A LIBRARY OF SOFTWARE PROGRAMS FOR THE IBM PERSONAL COMPUTER

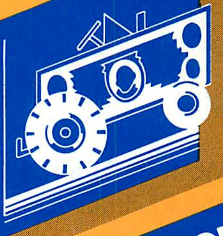
PC DISK MAGAZINE

VOL. 1 NO. 2

$29.95

USER MANUAL

LEASE/BUY COMPARATOR: COMPUTES CASH FLOW EQUIVALENCES FOR CAPITAL ACQUISITIONS



PERSONAL CASH FLOW MANAGER—PART TWO: INTEGRATED MANAGEMENT OF ALL CASH ACCOUNTS

BASIC CROSS REFERENCE: A COMPREHENSIVE INDEX OF PROGRAM ELEMENTS

2 DISK UTILITIES: A COMPLEMENTARY COMBINATION OF DISKETTE AND FILE MANAGEMENT TOOLS

PANGO: ARCADE-STYLE ACTION ON A SHIFTING ICE MAZE

CONTENTS

A LIBRARY OF SOFTWARE PROGRAMS FOR THE IBM PERSONAL COMPUTER

PC DISK MAGAZINE

PUBLISHER
André M.J. Van Hattum

EDITORIAL
Editor: Morris Effron
Contributing Editors: Peter Norton, Gary Oppenheimer, Mark Zachmann
Assistant Editors: Jennifer de Jong, Eric Marcus

ART & DESIGN
Cristina Botta, Robert Conrad, Don Menell Directions

CIRCULATION
Subscription Director: Chet Klimuszko
New Business Manager: Eric A. Bernhard
Renewal and Billing Manager: Shane Boel
Direct Retail Sales: Lynn Kujawa

CONSUMER COMPUTER AND ELECTRONICS MAGAZINE DIVISION
President: Larry Sporn
Vice President, Marketing: J. Scott Briggs
Vice President, Circulation: Carole Mandel
Vice President, General Manager: Eileen G. Markowitz
Creative Director: Peter J. Blank
Editorial Director: Jonathan D. Lazarus
Promotion Manager: Ronni Sonnenberg

ZIFF-DAVIS PUBLISHING COMPANY
President: Richard P. Friese *President, Consumer Magazine Division:* Albert S. Traina *Executive Vice President, Marketing and Circulation:* Paul Chook *Senior Vice Presidents:* Philip T. Heffeman, Sidney Holtz, Edward D. Muhlfield, Philip Sine *Vice Presidents:* Baird Davis, George Morrissey *Treasurer:* Selwyn Taubman *Secretary:* Bertram A. Abrams

PC Disk Magazine is published 9 times per year, by Ziff-Davis Publishing Company, One Park Avenue, New York, N.Y. 10016. Subscriptions are available at $179.00 for 6 issues (residents of CA, CO, CT, DC, FL, IL, MA, MD, VT add applicable tax). POSTMASTER: send address changes to *PC Disk Magazine*, 39 Hanover Avenue, Morris Plains, NJ 07950.
Editorial and Business Office: One Park Ave., New York, NY 10016. Editorial (212) 725-3500. For subscription inquiries and service write to *PC Disk Magazine*, 39 East Hanover Ave., Morris Plains, NJ 07950.
PC Disk Magazine is an independent journal, not affiliated in any way with International Business Machines Corporation. IBM is a registered trademark of International Business Machines Corp. Entire contents copyright © 1983 Ziff-Davis Publishing Company. All rights reserved; reproduction in whole or in part without permission is prohibited.
ISBN # 0-87165-102-5 PC DISK MAGAZINE VOLUME 1 NUMBER 2

FROM THE EDITOR

This second issue of *PC Disk Magazine* reaffirms our commitment to regularly bring you software that is both useful and of high quality. You might say that at the price we charge for the magazine, we have an obligation to do so. We agree. We believe that there is a serious need for software that expands the utility of your PC through diversity and variety. You will get more value from your PC with a collection of programs to perform a variety of useful tasks. People use computers quite differently, and those uses are constantly evolving. Our charter is to provide programs that do many different things, that are reliable, well-documented and instructive, thereby offering PC owners the greatest value.

Our program offerings in this issue are particularly diverse. In the business category, the *Lease/Buy Comparator* offers a sophisticated analysis of an important financial decision: whether or not to make a costly acquisition, and further, whether to purchase or lease the item. The *Confidant* offers a very different kind of business assistance. It will protect your private information by encoding that information quickly and easily.

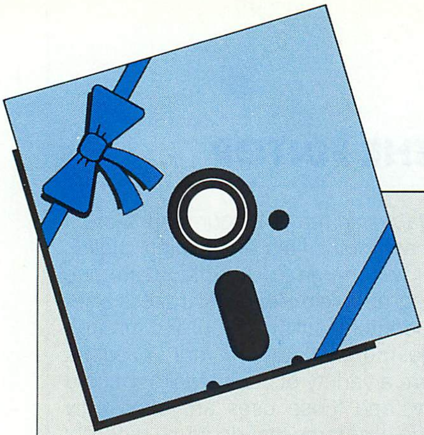
The computer tool box is extremely well-stocked in this issue as well. The *BASIC Cross Reference* is an invaluable aid to any significant BASIC programming effort. The experienced programmer knows the utility of such a program reference, while the novice programmer will benefit from learning to use such a tool. *Disk Label* and *Disk Inventory* together form an easy-to-use yet complete software library system for all your diskettes. We think the design and performance of this system are particularly good.

Then of course there's Part II of the *Personal Cash Flow Manager,* which expands this already thorough checking account monitor into a comprehensive cash tracking system. With Part II your financial activity can be budgeted, recorded, verified, archived and reported. It does everything but write the checks.

We don't expect all these programs to appeal to all our readers; that's almost an assumption of our publishing philosophy. We do require however, that they all be well-conceived and well-executed.

Let us know what you think, and let us know if you have any software that we might consider for publication. Our open submission policy can make *PC Disk Magazine* both practical and profitable for you.

Morris Effron



EXPAND YOUR SOFTWARE LIBRARY

This issue of *PC Disk Magazine* has already increased your software library for your IBM personal computer. If you haven't subscribed to *PC Disk Magazine*, just think of how easy it will be to expand your library with future issues delivered to your door at substantial savings. And if you already are a subscriber, wouldn't you like to share your knowledge with a friend and get them a gift subscription to *PC Disk Magazine*.

Either way, you can save $60 with this introductory offer and give yourself or a friend up to 72 programs for as little as $2.00 each! *PC Disk Magazine* is *the* solution to the problem of building and maintaining an up-to-date library of tested software programs.

You will get up to 12 varied and ready-to-use programs and files in every issue—complete with an illustrated user manual—for the ultimate in ease, efficiency and economy.

Every program in *PC Disk Magazine* is developed by experts, tested by the editors of *PC Disk Magazine* and published by the people who bring you PC, guaranteeing quality and satisfaction.

Use the enclosed card to subscribe today or to give a friend a gift. If the card is missing, please write to:
PC Disk Magazine, CN 1916, Morristown, NJ 07960
For even faster service, call our toll-free number below and use your American Express, MasterCard or Visa . . . today!

**CALL
1-800-526-0666
for FAST service
and get
6 issues of
PC Disk Magazine
for only $119
a savings of 34%!**

TECHNICAL PREFACE

To help our readers make the most of PC Disk Magazine, *we would like to provide some background information concerning the editorial diskette, the accompanying manual, and how to use both. We don't expect all of the following topics to be of interest to all our readers. Nevertheless we have preferred to err on the side of comprehensive support, rather than leave any of our readers confused or bewildered. So we encourage everyone to at least skim this section to assure a solid background for the use of* PC Disk Magazine.

USAGE REQUIREMENTS

PC Disk Magazine has been designed for use on an IBM Personal Computer with a minimal set of hardware components: a keyboard, a monitor, and the PC itself with at least 64K of main memory. The display unit can be a monochrome display adapter and monitor, or the color graphics display adapter with either a color monitor or an RF Modulator and TV set. The computer itself can be the PC or the PC-XT.

These three pieces of equipment are all you need to run the majority of *PC Disk Magazine* software. Wherever possible we try to make the use of any other hardware optional. So, for example, many of the programs will generate printed output, but a printer is not required to use them. Occasionally however, due to the nature of a program or its design, a particular piece of equipment will be necessary. There are two such programs in this issue: *''PANGO''* and *''DOUBLE VISION''* both require a color display unit. When a program requires a piece of equipment not in the minimum configuration stated above, this component will be listed as a ''Special Requirement'' on the program's title page in this manual.

In regard to software, all *PC Disk Magazine* programs are designed to run under DOS 1.1 and DOS 2.0. Furthermore all BASIC programs in the magazine are designed to run under Microsoft's Advanced BASIC. Neither DOS nor Advanced BASIC are provided on the *PC Disk Magazine* diskette; they must be acquired separately. As a rule these are the only outside software elements you will need to use *PC Disk Magazine*. We will occasionally publish a program which uses some additional, publicly available software product. We will more commonly publish a program which is part of a series and therefore requires a program from the preceding issue of *PC Disk Magazine*. Part II of the *"Personal Cash Flow Manager"* in this issue is a case in point. In either case, any such additional software will be listed as a "Special Requirement" on the program's title page in this manual.

A closing remark on this topic is not so much a requirement as a recommendation. We recommend that you make a copy of your *PC Disk Magazine* diskette to work with, and save the original as a backup. In some cases, such as the *"Personal Cash Flow Manager"* in this issue, you will have to make a copy of the program in order to use it. The reason is that some programs create additional files as they run, and these files must be stored on diskette as well. You may have noticed that your *PC Disk Magazine* diskette is write-protected. Thus it cannot receive these additional files. So a separate, working copy is needed. These situations will be explicitly mentioned in the manual. In general though, where the manual refers to "your *PC Disk Magazine* diskette" you should read "your working copy of the *PC Disk Magazine* diskette."

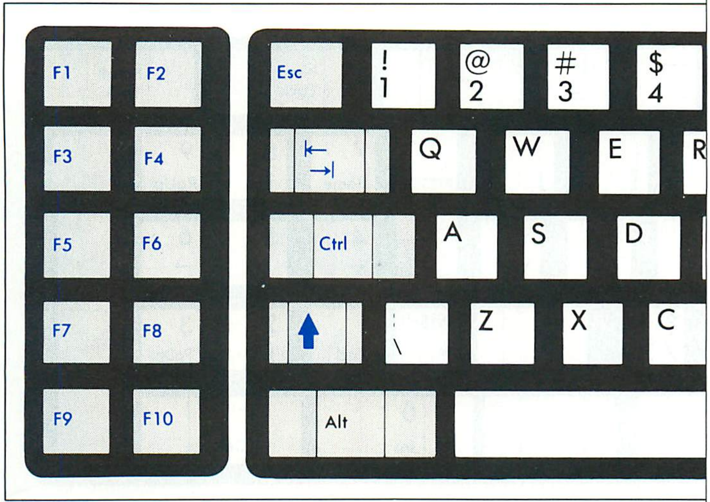
THE IBM PC KEYBOARD

In *PC Disk Magazine* we have tried to make our instructions as clear as possible by the consistent use of special key symbols. In addition to all the common typewriter keys, which we print as they would appear when typed, the IBM PC keyboard has a number of special keys. We have designed symbols for these keys, which are intended to resemble as much as possible the keys themselves. Since these symbols are used extensively throughout the instructions, we felt the following road map and glossary would help you, our reader, get any needed bearings.

PROBLEM HANDLING

We try our best to thoroughly test all *PC Disk Magazine* software, and provide instructions that cover all aspects of its use. Nevertheless, error-free software and exhaustive documentation are elusive goals. So if you have a problem, please contact us and let us help. Although we hope you will not need it, the address to write to is:

PC Disk Magazine
Department 741
One Park Avenue
New York, NY 10016



F6 THE FUNCTION KEYS

There are ten special keys called function keys located at the far left of the keyboard. They are numbered from F1 to F10. This stands for Function One, Function Two etc. These keys are often used to make single keystroke choices or commands.

Esc THE ESCAPE KEY

The ESC key is used most often for exactly what its name implies, to escape (exit) from various functions and processes.

Ctrl THE CONTROL KEY

This key is always used in conjunction with another key by pressing this key and the other key simultaneously. The purposes of the Control key vary widely depending on the application program.

|← →| THE TAB KEY

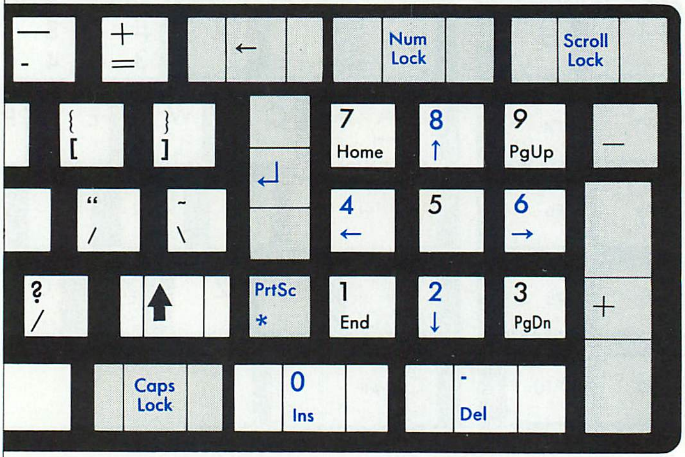
This key is commonly used for horizontal tabbing in text files. It is sometimes used by programs to allow rapid cursor movement during full-screen data entry.

← THE BACKSPACE KEY

The Backspace key is used to correct typing errors. By simply pressing the key the preceding character is erased and a new character can be entered.

⬆ THE SHIFT KEY

The Shift key is actually located on each side of the keyboard. It is used in conjunction with other keys to capitalize letters, get special symbols like: ! @ # $ % % * () and other special functions.



PrtSc * THE PRINT SCREEN KEY

This key is used with the Shift key to get a printout of exactly what is on the screen. In computer lingo this is called a screen dump, a dump of all the information on the screen to the printer. In *PC Disk Magazine* we also refer to this capability as "The IBM Print Screen Facility."

↵ THE ENTER KEY

This is the most used key on the keyboard. Most every time you need to give information to the computer you have to press this key to ENTER that information. This key can also be thought of as the carriage return, since it works similarly to the RETURN key on a typewriter.

Num Lock THE NUM LOCK KEY

Much like the Shift key, this key controls whether the numeric keypad will print numbers or act as cursor movement controls. You can tell that the NUM LOCK is on by pressing any of the number keys to see what is displayed on your screen. If it is the number, then the NUM LOCK is off. If it isn't, then NUM LOCK is on. To change the setting press NUM LOCK once.

2 ↓ 4 ← 6 → 8 ↑ THE CURSOR CONTROL KEYS

These are the arrows that point up, down and to each side. The NUM LOCK has to be on for these keys to be functional. These keys control cursor movement within some *PC DISK MAGAZINE* programs. They will move the cursor in the direction of the arrow.

0 Ins * Del THE INSERT AND DELETE KEYS

These keys really mean the INSERT and DELETE keys. And that is exactly how they are used. INS is used to insert new information and DEL is used to delete unwanted information. They are commonly used when editing BASIC programs, and can often be used when running BASIC programs as well.

Caps Lock THE CAPS LOCK KEY

This key is used to save you from having to hold the shift key down all the time to get capital letters.

Ctrl Scroll Lock THE CONTROL AND SCROLL LOCK KEYS

This key combination deserves special mention because of its importance in BASIC, the language of most *PC DISK MAGAZINE* software. These keys used together will interrupt the processing of any BASIC program. The keys should be used with caution because some interruptions can require you to start an entire procedure from the beginning.

TERMINOLOGY

In the preceding section we identified the special key symbols used in this manual, and gave a name to each one. For example:

↵

is called the Enter key. In our instructional narrative, it sometimes makes more sense to refer to a special key by its name rather than its symbol. Thus the key names in the preceding section are also special terms for the purposes of this manual. Familiarize yourself with the names to facilitate your use of the manual, and refer to the preceding section as a glossary of key names when necessary.

In addition to the key names, a few other terms and phrases are used in this manual that may be unfamiliar to you.

We commonly speak of putting a diskette in the "default drive." This may seem like a needlessly vague phrase. After all, we know a diskette drive always has a one letter identifier associated with it, so why not refer explicitly to that letter? The problem with using an explicit letter reference is that it can create confusion about what exactly you must do. In other words, operationally it does not matter whether you put the diskette in the A Drive, the B Drive or even the C Drive (if you have a third diskette drive). What matters is that you put the diskette *in the drive that is currently active*, i.e. the drive whose letter prompt currently appears on the screen. This is your "default drive" because any disk command without a drive letter will look at the diskette in this active drive. So when you put a diskette in the "default drive," you can then issue commands referencing that diskette without the use of letter identifiers.

Every start-up procedure for a BASIC program requires you to "Load Advanced BASIC into your PC." To run a *PC Disk Magazine* BASIC program, the BASIC Interpreter must be up and running on your machine—you must be "in BASIC." BASIC is really a program like any other. To start it you must load it from a disk into your PC and start it running. This is precisely what happens when you put your DOS diskette (or any diskette with the file BASICA.COM) in the default drive and type:

```
BASICA ↵
```

By so doing you "Load Advanced BASIC into your PC."

TEXT CONVENTIONS

Most of the textual conventions of this manual are fairly obvious. The use of special key symbols has been covered. The use of special key names in the narrative text has been discussed. That leaves two brief additional remarks concerning command lines.

The lines set apart from the narrative text, in bold print and a different color, are commands that should be typed in exactly as they appear. When two key symbols appear immediately next to each other in such a command line, they should be pressed simultaneously. For example:

```
⇧ PrtSc *
```

means press the Shift key and the Print Screen key simultaneously, thereby printing a copy of the current screen on your printer.

There is one exception to typing in command lines exactly as they appear. When a command includes a phrase such as "somename" or "programname" or "yourfile" you should replace that phrase (but not any punctuation) with a valid file name of your choice when you enter the command.

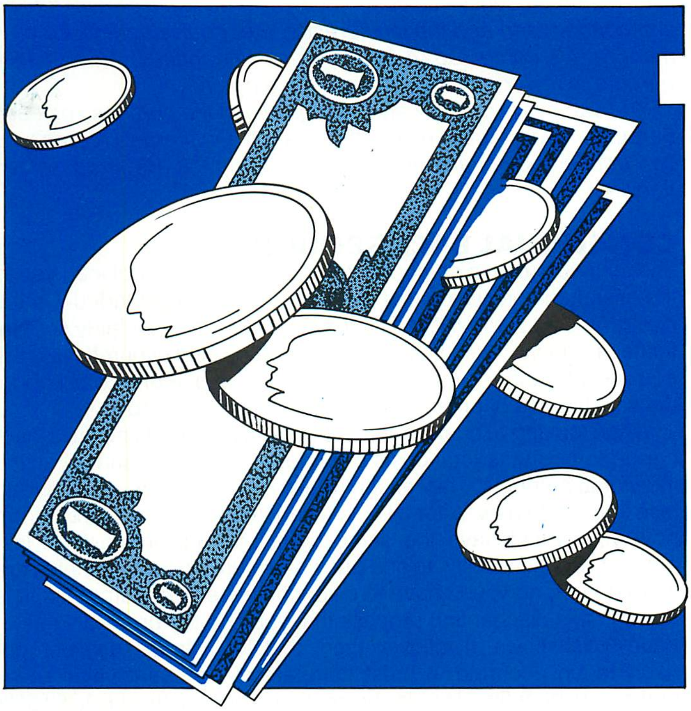
BACK ISSUES OF .... PC DISK MAGAZINE

Add to your *PC Disk Magazine* collection today, while copies are still available. The more complete your library of back issues, the more enjoyment and use you can get from your personal computer. The currently available back issue is Volume 1, Number 1. This offer is subject to availability. If this issue is out of stock, your payment will be refunded promptly.

Back issues of *PC Disk Magazine* are priced at $31.95 each, which includes postage and handling (residents of CA, CO, CT, DC, FL, IL, MA, MD, VT please add applicable tax). Please write to:

PC Disk Magazine
CN 1916
Morristown, N.J. 07960

Be sure to include the number of copies desired, and the address to which you would like those copies sent.



LEASE/BUY COMPARATOR

By Zyncon Inc.
Design by Jared Taylor

Special Requirements: None
Files Used: LEASEBUY.BAS

The acquisition of a new piece of equipment, such as a car, a computer, or a building, is one of the most fundamental financial decisions an individual or company has to make. The analysis that goes into making that decision is critical to personal and corporate well-being, and represents the foundation of the capital budgeting process. A substantial number of factors can be relevant to that decision, and each analysis will, in certain respects, be unique. One important subsidiary decision, which most every potential purchaser will have to make, is whether to lease or buy the equipment under consideration. The LEASE/BUY COMPARATOR *analyzes both the primary go/no-go acquisition decision and this subordinate question.*

The method used to compare lease versus buy alternatives is quite straightforward. First, determine the total cost of purchasing *the equipment. Assume that the purchase is financed by a loan. You fill in a screen of loan parameters and ownership benefits, from which the present value cost of purchase is then computed. This present value cost leads directly to the positive cash flow needed to justify the purchase. Here is the basis for our go/no-go acquisition decision. Then, calculate the lease payment amount that will result in the same present value cost. You fill in three lease parameters on a second screen and the indifference, (i.e, equivalent), lease payment rate is computed. A higher lease rate makes purchasing a bargain. A lower rate makes leasing a bargain.*

You can easily go back and change any of the purchase or lease parameters to examine different scenarios. Finally, as a record, you can print a copy of any of the screens using the IBM Print Screen Facility.

CONCEPTUAL BACKGROUND

The simplicity of this comparison procedure should not lead you to the mistaken conclusion that the analysis is simple-minded. On the contrary, the *LEASE/BUY COMPARATOR* employs a fairly sophisticated formula to calculate the present value cost of purchasing the equipment. Before explaining the formula used, a brief word about the significance of present value calculations is in order.

THE PROGRAM COMPARES PRESENT VALUES OF COSTS AND BENEFITS

When looking at costs or benefits over a period of time, it is inaccurate to compute total costs or benefits as just the sum of all the individual amounts. The reason is simple: $100 next year is worth less than $100 now. After all, if you had that $100 now, a year from now you would have the same $100 plus the interest on that amount. Consequently, to find out what future costs or benefits are worth to you now, you have to discount those future amounts. It is through this discounting process that you find the present value of future cash flows. It follows, then, that for a lease/buy comparison to be accurate, it must compare the present value of all costs and benefits. This is the methodology of the *LEASE/BUY COMPARATOR*.

There are four elements needed to compute the present value cost of a purchase: the present value of all the loan payments, the Investment Tax Credit (ITC), the present value of all depreciation, and the present value of the equipment's anticipated salvage value. Only the first term is actually a cost; the other three components are nonproductive benefits of equipment ownership (i.e., benefits not derived from the operation of the equipment). In discounting loan payments, depreciation, and salvage value, the discount factor used is the cost of capital that you enter.

The loan is assumed to be a standard commercial loan wherein principal is repaid in equal periodic installments, and interest is based on the outstanding principal balance in each period. Since interest payments are tax deductible, the interest amount in each period is multiplied by 1 minus the tax rate you enter, to reduce it appropriately.

The salvage value of the equipment is similarly multiplied by 1 minus the tax rate you enter, before being discounted.

Depreciation can be computed in either of two ways: straight-line or Accelerated Cost Recovery System (ACRS). Under straight-line depreciation, the depreciation amount in each year is constant, equaling the difference between the purchase price of the equipment and its salvage value, divided by the length of the depreciation recovery term in years. Under ACRS, valid depreciation recovery terms are 3, 5, 10, and 15 years. Each schedule has a table of percentages that determine the amount to be depreciated in each period. In either case the amount of depreciation in each period is multiplied by the tax rate you enter to adjust for its tax charge-off effect.

The ITC is not discounted, since its benefit is realized in the year that the purchase is made. The full ITC is 10 percent of the installed cost of the equipment. Under ACRS, the depreciation recovery term must be 5 years or more for the investment to use the full 10 percent. For a 3-year property, only 60 percent of the 10 percent may be claimed. Under straight-line depreciation, the recovery term must be 7 years or more to use the full 10 percent. For less than 3 years, no ITC is allowed; for 3 to 4 years, one-third of the 10 percent is allowed; for 5 to 6 years, two-thirds of the 10 percent is allowed.

Thus, you see the present value calculation for a purchase is not so simple. Determining the indifference lease payment rate is much easier, however. Once we know the present value of the cash flows associated with ownership, all we need to do is find the value of the annuity (representing periodic, identical lease payments) which, when discounted by the cost of capital already entered, will yield the same present value. The only wrinkle in this computation is that the percentage of the lease payment that is tax deductible, if any, must be factored into the equation so as to increase the amount of the indifference lease rate. This factor must be input by the user.

FIRST IS THE PURCHASE TERMS SCREEN

START-UP

To start the *LEASE/BUY COMPARATOR*, load Advanced Basic into your PC by typing:

```
BASICA ↵
```

Then put the *PC Disk Magazine* diskette in your default drive and type:

```
RUN "LEASEBUY ↵
```

PURCHASE SPECIFICATION

The first screen that appears is the purchase specification screen. On the left-hand side are the parameters required for the installment loan calculation. On the right-hand side are the factors affecting the benefits of ownership as well as the discount rate for the analysis (the Cost of Capital).

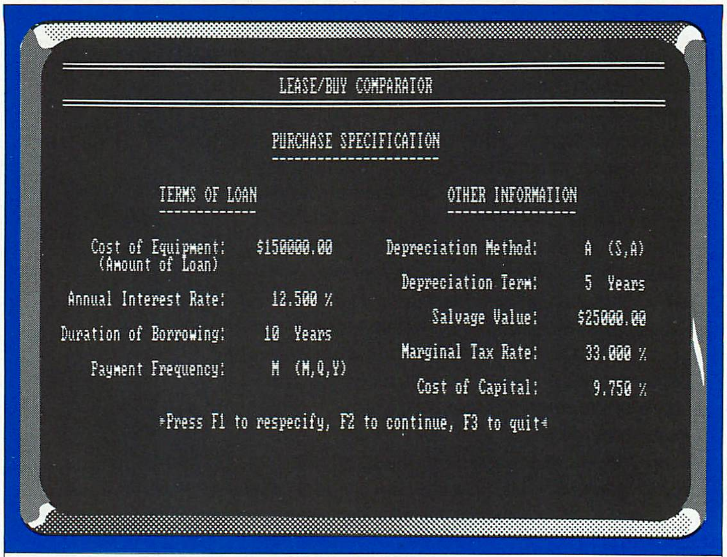
Certain default values will appear for different terms to indicate that values must be entered for these items. You fill in the screen by first completing the left-hand side items in sequence and then the right-hand side. Enter all the numbers without commas and with a decimal point if your entry is not an integer. Push:

```
↵
```

after every entry to go on to the next item. To leave the value for an item unchanged, type:

```
↵
```

when the cursor is at the beginning of that item. To replace a value, simply type in the new value. As soon as you start typing, you will



```
                              LEASE/BUY COMPARATOR

                             PURCHASE SPECIFICATION
                             ----------------------

              TERMS OF LOAN                            OTHER INFORMATION
              -------------                            -----------------

     Cost of Equipment:   $150000.00       Depreciation Method:       A  (S,A)
     (Amount of Loan)
                                           Depreciation Term:         5  Years
  Annual Interest Rate:      12.500 %
                                                Salvage Value:    $25000.00
 Duration of Borrowing:      10  Years
                                            Marginal Tax Rate:       33.000 %
     Payment Frequency:      M  (M,Q,Y)
                                              Cost of Capital:        9.750 %

               »Press F1 to respecify, F2 to continue, F3 to quit«
```

A complete specification of purchase terms.

WHEN PURCHASE TERMS ARE SET, THREE ACTIONS CAN FOLLOW

see the old value disappear. These specifications can only be entered in order. Therefore, if you want to change an earlier entry, you must cycle through all the other entries until you return to the one to be changed.

For Cost of Equipment enter any amount up to $100 million. The Annual Interest Rate for the loan can be any two-digit number with up to three decimal places. The Duration of Borrowing must be an integral number of years up to 99. The Payment Frequency for the loan—that is, how often payments are made—can be M for monthly, Q for quarterly, or Y for yearly.

The Depreciation Method can be S for Straight-line or A for ACRS. For Straight-line depreciation the Depreciation Term, which is the number of years over which the equipment is to depreciated, can be any integral number of years up to 99. For ACRS, the Depreciation Term must be 3, 5, 10 or 15 years. The Salvage Value of the equipment can be any amount up to $100 million. The Marginal Tax Rate, which is your effective tax rate, and the Cost of Capital, which is the discount rate for the analysis, can be any two-digit number with up to three decimal places.

When you have successfully completed the purchase specification screen, a message line will appear offering you three choices of action. To have the present value of the purchase calculated and continue to the next screen, press:

F2

To cycle through the purchase parameters again to make changes or corrections, press:

F1

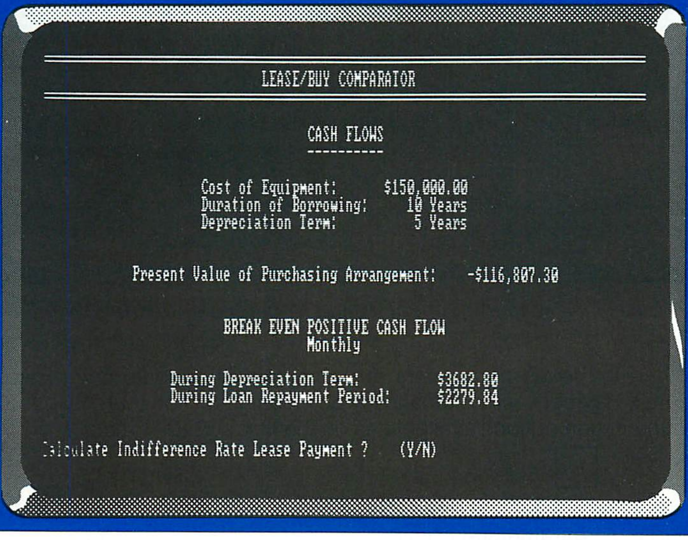
When you do so, the cursor will return to the first item, Cost of Equipment, allowing you to cycle through the list of parameters again. To end execution of the *LEASE/BUY COMPARATOR* and return to Advanced Basic, press:

F3

BREAK-EVEN CASH FLOW

When you choose to continue, a second screen will be displayed. This second screen presents an intermediate result in the lease versus buy comparison, but it is one that is basic to the capital budgeting decision. This intermediate result is the total present value cost of acquiring the equipment, followed by the periodic positive cash flow that must be generated by the use of this equipment in order to break even on the purchase. This latter result is the primary part of the capital budgeting decision: Does the return on the use of the equipment justify its purchase in the first place?

The periodicity of the break-even positive flow is the same as the payment frequency of the loan. You will notice, however, that there are two different time frames over which this break-even calculation has been done. The first cash flow amount applies to the depreciation recovery term, the second to the duration of borrowing for the loan. The *LEASE/BUY COMPARATOR* allows for these two different time frames, depreciation term and loan term, to accommodate more varied financing arrangements. As a consequence, you may want to consider your break-even cash flow over either length of time. The result appears here for your information and consideration.



```
                    LEASE/BUY COMPARATOR

                         CASH FLOWS
                         ----------

              Cost of Equipment:        $150,000.00
              Duration of Borrowing:      10 Years
              Depreciation Term:           5 Years

       Present Value of Purchasing Arrangement:    -$116,807.30

                BREAK EVEN POSITIVE CASH FLOW
                           Monthly

            During Depreciation Term:          $3682.80
            During Loan Repayment Period:      $2279.84

Calculate Indifference Rate Lease Payment ?   (Y/N)
```

The critical cash flows based on given purchase terms.

At the bottom of this screen you are asked whether you want to go on to calculate the indifference rate lease payment based on this purchase arrangement, or not. If you type:

N

you will be returned to the first screen. If you type:

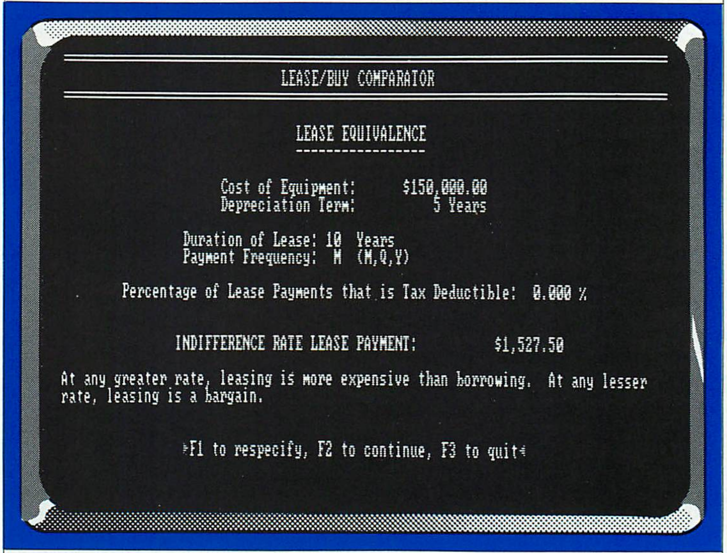
Y

you will proceed to the third and final screen, where you will specify your lease terms, and the indifference rate will be calculated.

LEASE EQUIVALENCE

THE EQUIVALENT LEASE PAYMENT IS DETERMINED LAST

The third screen displays the previously entered Cost of Equipment and Depreciation Term at the top for your reference. In most cases you will probably want to use the Depreciation Term as the Duration of Lease, if the depreciation term truly represents the useful life of the equipment. (You lease the equipment for the duration of its useful life.) Nevertheless, you may enter a lease duration of your choice, up to 99 years. The Payment Frequency, like the loan, may be M for monthly, Q for quarterly, or Y for yearly. The tax deductible portion of lease payments may be any two-digit number with up to three decimal places.



```
                         LEASE/BUY COMPARATOR

                           LEASE EQUIVALENCE
                           -----------------

                    Cost of Equipment:      $150,000.00
                    Depreciation Term:          5 Years

                Duration of Lease: 10  Years
                Payment Frequency:  M  (M,Q,Y)

          Percentage of Lease Payments that is Tax Deductible:  0.000 %

              INDIFFERENCE RATE LEASE PAYMENT:           $1,527.50

At any greater rate, leasing is more expensive than borrowing.  At any lesser
rate, leasing is a bargain.

                  ▸F1 to respecify, F2 to continue, F3 to quit◂
```

A lease payment amount equivalent to purchase (the indifference rate).

When the lease parameters are complete, the Indifference Rate Lease Payment will automatically be calculated. You then have the same action options available as on the first screen. Press:

F2

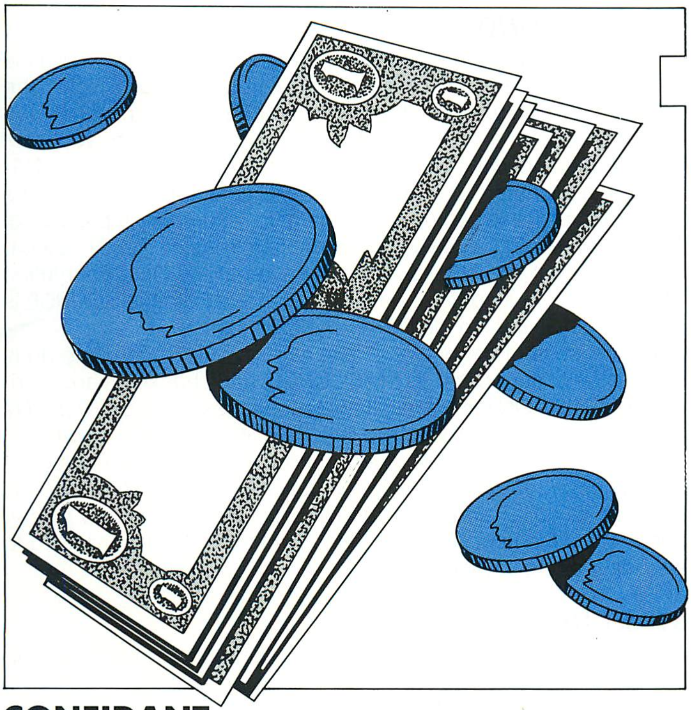
to continue, that is, to return to the first screen for revisions to your purchase parameters. Press:

F1

to respecify your lease parameters, that is, to cycle through the three parameters on this third screen again. Lastly, press:

F3

to end the *LEASE/BUY COMPARATOR*.



CONFIDANT
By Data Sage Inc.

Special Requirements: None
Files Used: CONFIDE.EXE

Whether you're a social scientist on the verge of a major breakthrough, a politician planning a new policy statement, or a grandmother protecting a secret recipe for stuffed cabbage. THE CONFIDANT *will be a priceless addition to your software library. With it you can encode sensitive data, prevent undetected alteration of your data, and protect vital information transmitted over communication lines.*

THE CONFIDANT *will take your sensitive data, encrypt it, and then guard access to the information through the use of a password. The encryption process itself is based on the mathematical combination of a password that you specify with each element of your informa-*

tion. The result is a new string of characters for every element of your original information, an unintelligible sequence known as "cipher text." THE CONFIDANT *allows you to encode data, decode data, print it and erase it. Through the use of a single activity screen,* THE CONFIDANT *will successfully guide you through all of the steps necessary to make certain that your secrets remain secrets.*

YOUR ONLY COMMAND IS TO TRANSLATE INPUT TO OUTPUT

BACKGROUND

The encryption procedure itself is fairly straightforward. You tell the computer where your original text will be coming from (the SOURCE), which can be either an existing file or text typed in directly from the keyboard. Then you tell the computer where you want the text to go once it's been turned into secret code (the DESTINATION) - to a printer, or a new file for example. Lastly, to drive the translation process you have to dream up, and enter, a password. The computer will then take the information from the SOURCE location and turn it into code using a variation of the Data Encryption Standard (DES) algorithm. As the original text from the SOURCE is encrypted, it is sent to the DESTINATION location.

The process of decryption is just as easy. All you need to do is specify the encoded text as the SOURCE, and choose an appropriate DESTINATION for the resulting decoded text. *THE CONFIDANT* can tell automatically whether the SOURCE text is original text (uncoded) or cipher text (coded), and thereby knows whether to encode or decode it. Your only instruction to *THE CONFIDANT* is to translate SOURCE to DESTINATION. The essential key to proper decoding is the specification of the exact same password used to encode the original text.

START UP

To start *THE CONFIDANT*, place your PC Disk Magazine diskette in the default drive and type:

```
CONFIDE ↵
```

An introductory screen will appear followed by the main activity screen. This is the screen where you'll enter the SOURCE and DESTINATION of your text as well as the PASSWORD. The three boxes displayed on the right side of the screen are where this information is to be entered.

At the bottom of the screen are eight activities, each associated with a different function key: ENcipher or DEcipher, Copy, Flip, Hide, Directory, Erase SOURCE, HELP, and END. Let's first have a look at the encryption process itself for which we'll use the F3 function key. F3 is used for both encryption and decryption since *THE CONFIDANT* is programmed to determine whether the data in the SOURCE should be encoded or decoded.

INPUT/OUTPUT

Before encryption can begin, it's necessary to fill in the three information fields. When you begin the program, the first box, SOURCE, is highlighted. Your first task is to tell the computer where the text to process will be coming from. Your SOURCE can either be an existing file or you can type in text directly from the keyboard (CON-

SOLE). The computer selects CONSOLE by default. To leave the value in any one of the three boxes unchanged, just press:

↵

when that box is highlighted. Otherwise type in the value you want followed by the Enter key. The Insert and Delete keys are available to help you edit the contents of a box more quickly. For our example let's enter our original text directly from the keyboard by selecting CONSOLE as our SOURCE. Since this already appears as our SOURCE, just press:

↵

Once you complete your SOURCE selection, the next box, DESTINATION, will be highlighted.

SPECIFY INPUT, OUTPUT, AND PASSWORD

The DESTINATION tells THE CONFIDANT where to send the processed text. The DESTINATION may be the CONSOLE (i.e. your terminal), a file, a (parallel) printer or a SERIAL port (most commonly selected for communications transmission). Let's send the text to a file called SECRET. Type:

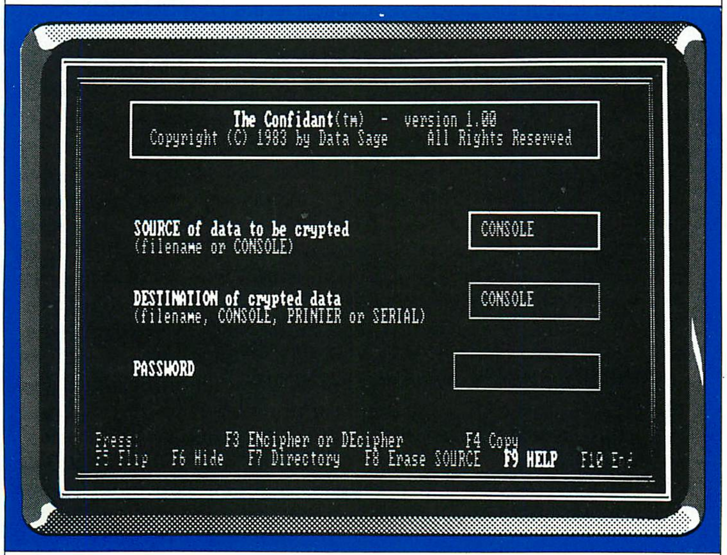
SECRET

If this file already exists, the computer will write over it without asking, so be sure this is a new file. Once you've finished entering the file name, press:

↵

and the PASSWORD box will be highlighted.

The most complicated step in choosing a password is committing it to memory, or writing it down. If you forget your password, you're in the same boat as anyone else who wants to decrypt your text. You simply won't be able to do it.



```
          The Confidant(tm)  -  version 1.00
   Copyright (C) 1983 by Data Sage    All Rights Reserved

SOURCE of data to be crypted                      CONSOLE
(filename or CONSOLE)

DESTINATION of crypted data                       CONSOLE
(filename, CONSOLE, PRINTER or SERIAL)

PASSWORD

Press:            F3 ENcipher or DEcipher         F4 Copy
F5 Flip   F6 Hide   F7 Directory   F8 Erase SOURCE   F9 HELP   F10 End
```

The main activity screen at start-up.

You should choose a password as close to sixteen characters as possible. The longer the password, the more secure the encryption. You can use upper or lower case letters, but recognize that *THE CONFIDANT* does distinguish between the two. Enter a password and press:

↵

ENCRYPTION

To begin encryption, press:

F3

A new screen will appear with one box marked SOURCE. Type in some sample text, pressing the Enter key at the end of each line as you would normally. When you press the Enter key, the text just typed will be encoded. You will not see the encoded text, but the incrementing block number at the bottom of the screen will let you know that encryption is under way. When incrementation stops, your text has been processed. After you've finished entering a few sample sentences, press:

F10

to end the encryption procedure. The message "The Confidant finished . . . press any key to return to Main Menu," will appear at the bottom of the screen. Hit any key, and you'll be returned to the main activity screen.

VIEWING TEXT

The COPY and FLIP functions are often used in conjunction when you want to view text. The COPY function is used to take text in the SOURCE and copy it into the DESTINATION for viewing. It simply moves the text in the SOURCE location to the DESTINATION location, making no changes to that text in the process. For example, if you've just encrypted text and want to see it displayed in coded form, you would use the COPY function to display the encrypted file on the CONSOLE or send it to a printer.

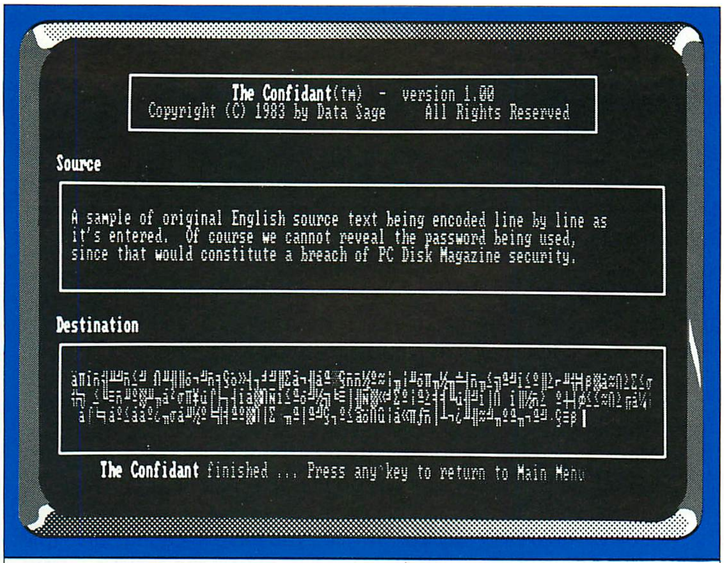
USE COPY TO DISPLAY A FILE AS IS

Frequently you will use the CONSOLE to enter text that will be encrypted, and have the result written to a disk file. Afterward, you will probably want to examine the encrypted text file by either COPYing or decrypting the text in that file back to the CONSOLE. By so doing, you can verify that the encryption was successful.

This is where the FLIP function comes in. Without the FLIP function you would have to type the name of the file just encrypted over CONSOLE in the SOURCE box, and the word CONSOLE over the file name in DESTINATION. The FLIP function accomplishes these actions in one move by switching the SOURCE and DESTINATION entries.

Let's try this procedure. Following the encryption procedure, you will have CONSOLE in the SOURCE box and SECRET in the DESTINATION box. To view the encrypted text, we first have to flip SOURCE and DESTINATION. Press:

F5


```
The Confidant(tm)  -  version 1.00
Copyright (C) 1983 by Data Sage     All Rights Reserved

Source

A sample of original English source text being encoded line by line as
it's entered.  Of course we cannot reveal the password being used,
since that would constitute a breach of PC Disk Magazine security.

Destination

[illegible]

The Confidant finished ... Press any key to return to Main Menu
```

Console to Console encryption (PC pocus).

and the entries in the SOURCE and DESTINATION boxes exchange places.
Then press:

F4

the COPY function, to look at what is in your encrypted file, SECRET.

When you press the F4 function key, the screen with the single rectangle will appear. In that box will appear the encoded file contents (it'll look like nonsense, but that's how it's supposed to look). You can interrupt this copying procedure at any time by pressing:

F10

Just for fun, if you want to view the encryption as it's taking place, put CONSOLE in both the SOURCE and DESTINATION boxes, enter a password and press function key F3 for encryption. A form will appear on the screen with two large rectangular boxes stretching across the screen. The top box will be labeled SOURCE and the bottom one DESTINATION. Type in sample text, and it will appear in the box labeled SOURCE. When the line if full, press:

↵

In the destination box a little chunk of nonsense text will start to appear. As you type in the English text, the computer will encrypt it and display the encrypted result in the DESTINATION box.

FLIP SWITCHES THE INPUT AND OUTPUT ENTRIES

DECRYPTION

To decrypt your encoded text, all you have to do is enter the name of the file containing the encoded text in the SOURCE box, and designate a DESTINATION. For this exercise, let's choose the CON-

SOLE as the DESTINATION. In the SOURCE box enter the file name SECRET, enter CONSOLE in the DESTINATION box, and then enter the password for this particular file. Then press:

F3

and the screen with the single rectangular box will appear. In that rectangular box you will see the decrypted contents of the file SECRET scroll by.

PRESS F9 FOR HELP

OPTIONS

If at any point you get lost and can't figure out what you're doing, try the HELP function, F9. It may provide some answers.

The HIDE function allows you to hide the information in the three boxes on the main activity screen. Let's say an employee walks into your office while you're encoding some very sensitive company information. Press the F6 function key and the SOURCE, DESTINATION and PASSWORD fields will go blank. Press F6 again and the information will reappear in the boxes.

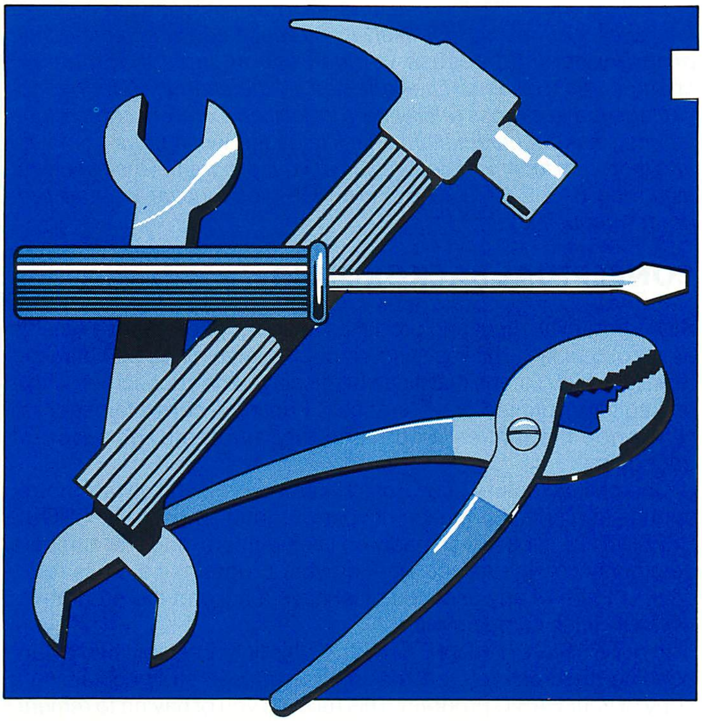
The F7 DIRECTORY key lists the files you have in case you've forgotten the name of a file. It will find the file listed in the SOURCE box, or if the SOURCE box is empty or is filled with the word CONSOLE, it will list all the files on the default disk.

F8, the ERASE key, allows you to erase the file listed in the SOURCE box. For example, if you've encrypted a source file onto a new file, you can erase the original English text file by pushing function key F8. To avoid an accidental erasure, the program asks you to confirm your ERASE command by pressing F8 a second time.

Both F7 and F8 allow the use of the "wildcard" characters, "?" and "*". If you are not familiar with the wildcard conventions of DOS, please refer to page 3-9 of your DOS 1.1 manual or 2-19 of your DOS 2.0 manual.

The END key, F10, will stop *THE CONFIDANT* at any time and return you to the place at which you gave the last command. By pressing the F10 key from one to four times, depending on what *THE CONFIDANT* is doing, you can end *THE CONFIDANT* completely and return to the operating system.

PROGRAMMING TOOLS



BASIC CROSS REFERENCE

By Zyncon Inc.

Special Requirements: None
(Works with Microsoft BASIC and Advanced BASIC, versions 2.0 and earlier).
Files Used: BASXREF.EXE

Programming is by nature a frustratingly precise effort, and programming in BASIC is no exception. It's hard enough to develop a sound, logical solution to your problem, but the most difficult part is the debugging. Dealing with unidentified line numbers, strange variable values and erroneous results can really drive you up a wall. And then there's the so-called minor change to a working program that causes it not to work properly anymore. Even when your program finally works you're not finished—the program must be documented. It can all be enough to make you wish you never started. Unless of course, you happen to have the BASIC CROSS REFERENCE.

The idea behind a cross reference is simple. It is a catalog, in the form of an index, of all the key elements of your program. It contains a list of all the variables in your program, along with the line numbers where those variables are used. All references to line numbers in your program are also recorded, along with the line numbers where the reference occurs. And there's more. Armed with a comprehensive index like this one, tracking down problems becomes a lot easier. You're also less likely to introduce new problems during program editing because you can see all lines that must be changed, as well as all lines affected by the change. A further advantage is that a cross reference is in itself an excellent cornerstone for program documentation; it facilitates the writing of a complete and thorough technical description of your program. Once you start using the BASIC CROSS REFERENCE, *you'll never want to program without it again.*

CONCEPTUAL BACKGROUND

The *BASIC CROSS REFERENCE* is a program that takes your BASIC program as input and produces an index of program elements as an output file. The resulting output file is an all ASCII character text file, which means that it can be displayed on your screen with a text editor or printed on your printer. Printing a copy of your cross reference file makes it conveniently available for program debugging.

THE *BASIC CROSS REFERENCE* PRODUCES AN INDEX OF PROGRAM ELEMENTS

One critical feature of a Cross Reference is that it always corresponds to a specific version of your program. The BASIC CROSS REFERENCE is based primarily on line numbers, and line numbers commonly change as you work on your program. If a cross reference is to be of any use, you must have a copy of the program to which its lines numbers refer.

In our *BASIC CROSS REFERENCE* this link is established by appending the cross reference section to a file already containing a copy of your BASIC program. This relieves you of having to remember which cross references go with which program versions. It requires, however, that you make a copy of your BASIC program before running the *BASIC CROSS REFERENCE.*

Not just any copy of your program will do, however. BASIC program files come in two varieties: tokenized and ASCII. Briefly stated, the tokenized form of a BASIC program has all the instructions stored in a condensed form, which the computer can interpret but we humans can't (not without tremendous effort, anyway). The ASCII form of a BASIC program contains all the instructions in English, as they were typed into the computer. When you are working in BASIC, you are using the tokenized form of your program, even though it appears as complete English text (BASIC automatically translates between the two forms as you work).

The *BASIC CROSS REFERENCE* takes advantage of these two forms of the exact same program. It uses the tokenized form of your program for indexing, because indexing can be done much faster using this form. The results of the indexing are appended to an ASCII copy of your program, so that the resulting file can be displayed or printed in its entirety—program and cross reference both. Therefore, you must create an ASCII version of your program before you run the *BASIC CROSS REFERENCE.*

START UP

The program to be cross referenced, and the ASCII copy of that program, should both be on a working diskette. To create the ASCII copy of your program, load Advanced BASIC into your PC by typing:

```
BASICA ↵
```

Then put the working diskette with a copy of your BASIC program in the default drive and type:

```
LOAD "programname ↵

SAVE "programname.ASC",A ↵
```

The second command will cause an ASCII version of your program to be written to the working diskette. To invoke the *BASIC CROSS REFERENCE* you must exit BASIC by typing:

```
SYSTEM ↵
```

If you have only one disk drive you will have to copy the *BASIC CROSS REFERENCE* program, BASXREF.EXE, onto your working diskette. Otherwise, you can put the *PC Disk Magazine* diskette in your default drive and your working diskette in a second drive. In either case, begin execution of the *BASIC Cross Reference* by typing:

```
BASXREF ↵
```

OPERATION

You are first asked to enter the name of the BASIC program to be cross referenced. If you enter a filename with no extension, the extension is assumed to be .BAS. All the standard DOS file naming conventions apply here, including the use of a drive letter prefix. You are then asked to enter the name of the result file, that is, the file that currently contains just the ASCII version of your program. If you enter just a filename, an extension of .ASC is assumed. If you enter nothing and simply press the Enter key, the program will use the filename of your first response with an extension of .ASC for the output filename. This is the default shown in square brackets with this prompt. For example, if you followed the procedure above to make the ASCII version of your program, you would type:

```
programname ↵
```

in response to the first question, and:

```
↵
```

in response to the second.

If both your input and output files are the right type, cross referencing will begin. The program makes two passes through your tokenized BASIC file, compiling index information during each pass. The computer will tell you each line it is processing as it works. When the cross referencing is complete, the index compiled is appended to your ASCII source code, and a message to this effect is displayed. This completes the execution of the *BASIC CROSS REFERENCE*. If you want to cross reference another file, you must reinvoke the program.

YOU NEED ASCII AND TOKENIZED VERSIONS OF YOUR PROGRAM

THE RESULT IS A WEALTH OF PROGRAM FEATURES IN SEVERAL SECTIONS

RESULTS

Your output file now contains a wealth of information about your BASIC program. What follows is a description of each section of the cross reference file, in the order in which they appear. The section headings will always appear in your cross reference file preceded by three asterisks. The section itself may, however, be empty if there are no elements of this kind in your program.

First, of course, is the listing of your program, the result of your ASCII copy procedure. The program listing ends with three rows of asterisks containing the heading "CROSS REFERENCE SECTION," following which is the first section of the cross reference proper.

The first section lists undefined line numbers, which are really program errors. These are line numbers referenced in your program which do not exist. The nonexistent line number appears first, followed by the program line numbers wherein the nonexistent line is referenced. The lines containing the invalid reference must be corrected before your program will run. This first section is really the only error section. All other sections contain reference information.

The indexlike format of each section follows the same pattern. The program element referred to is flush with the left-hand margin. To its right the program line numbers in which that program element appears are given in ascending order.

After undefined line numbers is the section entitled "Variable list" which, not surprisingly, contains a list of all the variables of the program in alphabetical order. The variable names are exactly those used in your program—no truncation, no abbreviation. Arrays are shown with a single left parenthesis following the variable name.

Following the complete variable list is a separate section which lists variables appearing in a COMMON statement in your program. This section could be considered a subsection of the complete variable list, since the variables here also appear in that preceding section. They have been repeated as a separate section however, to make it easier for you to identify these variables that are critical to the chaining of programs. This section also provides the variable name key with which you can go back to the complete variable list and see where else these variables are used in your program. Only one line number follows the variable name, which is the line number containing the COMMON statement for this variable. Other references to this variable appear in the preceding section.

Functions are listed in the next section, also in alphabetical order. The format of this list is slightly different from the complete variable list. The function name appears at the left-hand margin as usual. However, the first line reference in the list is the line in which the function is defined, not necessarily the first line where the function appears. Following the definition line number are all the other line references in ascending order.

"Line number references" are the next section. Again the program element, i.e. the line number referred to, is at the left-hand margin, followed by the line numbers where references to it occur. The line numbers on the left-hand side of this section are those appearing after such keywords as GOTO, THEN, and ELSE.

There is one special case of line number referencing which is identified by a "+" sign appearing before the line number referred to. This is the case where the same line number is referred to in a direct branch statement, such as GOTO, THEN or ELSE, and also in a GOSUB statement. Technically speaking this is an error. If you branch directly to a line number which is the beginning of a subroutine, you will probably get the error "Return without GOSUB" when your program reaches the RETURN statement which marks the end of this subroutine. Alternatively, a subroutine branch to a statement which does not have a subsequent RETURN statement is sure to produce undesirable results.

THE INDEXLIKE FORMAT OF EACH SECTION FOLLOWS THE SAME PATTERN

There are nevertheless ways whereby one can get away with this kind of double duty branching. It is certainly not recommended practice, but it can not be unconditionally called an error. However, it is certainly something that a programmer should be alerted to, since there is a strong likelihood that it indicates an error. That is the motivation for having a special flag to identify such line references.

When you find such a line, your first action should be to peruse the next section, which contains a list of all subroutine entry points followed by all lines where the subroutine is invoked. The left-hand side of this section lists all line numbers appearing in a GOSUB statement. Here again line numbers also referenced by a direct branching statment are preceded by a "+" sign. If you see such an annotation, you should check the list of subroutine invocations here along with the list of direct branches in the previous section for a possible programming error.

The next section handles another special form of program branching: a line number following the RETURN keyword of a subroutine. This feature, allowing you to return from a subroutine to a specific line number, rather than the point of invocation, is only available in version 2.0 of Advanced BASIC. The format of this section is the line returned to flush left, followed by the line(s) containing the RETURN statement.

The next and last section of the *BASIC CROSS REFERENCE* is a list of files referenced in the program. This list is actually broken down into a series of subsections, corresponding to the different ways in which files can be referenced in BASIC. In all subsections the file name referenced appears at the left-hand margin, followed by the line numbers where the particular kind of file reference is made.

The first subsection contains a list of all files referenced in an

OPEN statement. These will commonly be data files to your program.

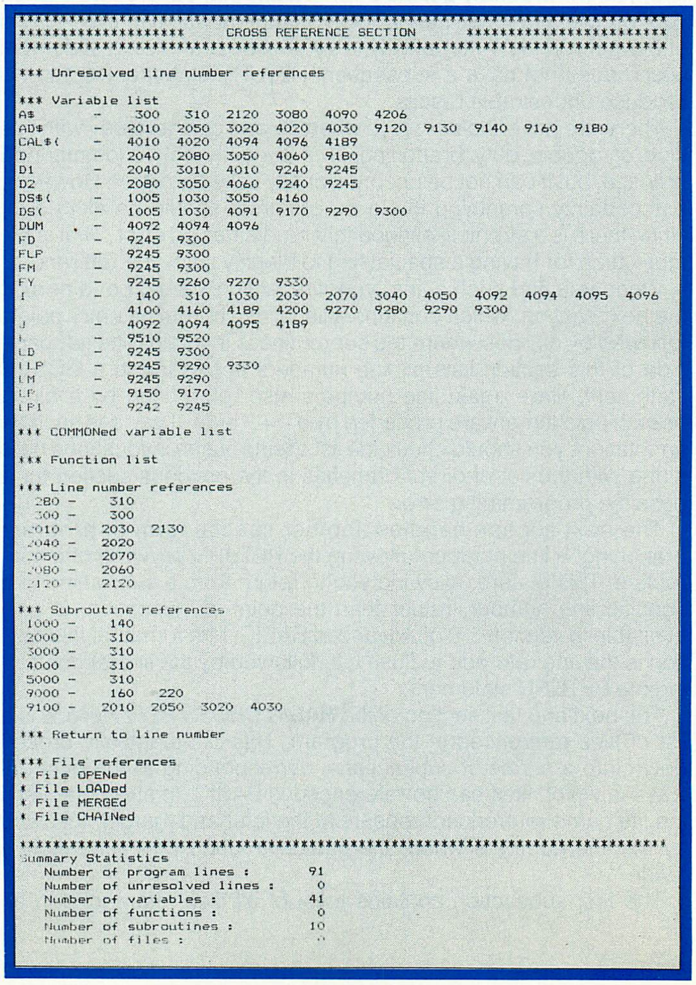
The next subsection lists all files appearing in a LOAD statement. This command effectively ends execution of your program, clearing all current BASIC and bringing in a new program. An optional parameter on the LOAD statement can start execution of the program loaded.

THE *BASIC CROSS REFERENCE* CLOSES WITH A BRIEF SET OF SUMMARY STATISTICS

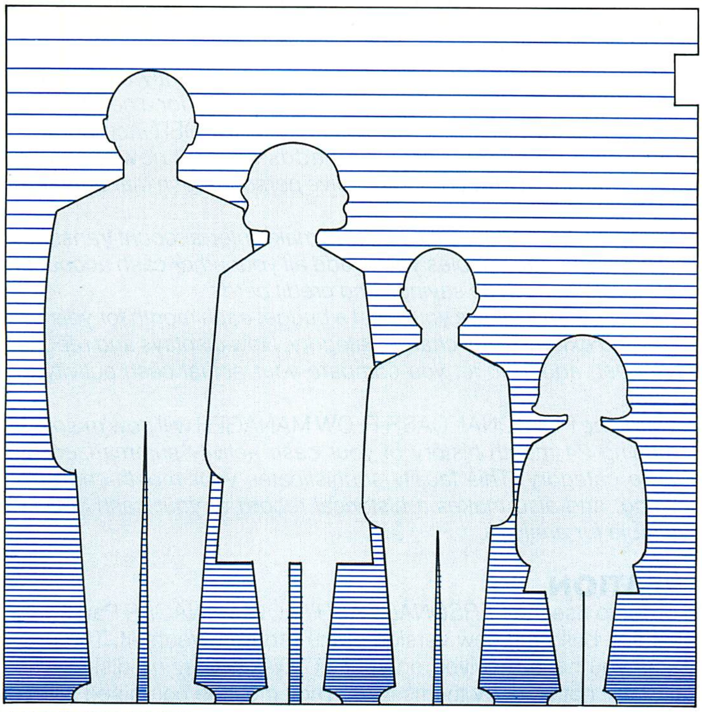
Next are files MERGEd. The MERGE statement combines the contents of the file referred to, which must be the ASCII version of a program file, with the program currently in memory. This is a nice way to dynamically modify the program executing.

Last is the subsection for the files CHAINed to. As with LOAD and MERGE, the file referenced in a CHAIN statement must be a program file. The distinguishing feature of chaining, by contrast with these other two program file operations, is that variables and open files can be preserved after a chaining operation occurs.

The *BASIC CROSS REFERENCE* closes with a brief set of summary statistics, which tally various program elements.



```
**************************************************************************************
********************      CROSS REFERENCE SECTION      ***********************
**************************************************************************************

*** Unresolved line number references

*** Variable list
A$       -    300   310  2120  3080  4090  4206
AD$      -   2010  2050  3020  4020  4030  9120  9130  9140  9160  9180
CAL$(    -   4010  4020  4094  4096  4189
D        -   2040  2080  3050  4060  9180
D1       -   2040  3010  4010  9240  9245
D2       -   2080  3050  4060  9240  9245
DS$(     -   1005  1030  3050  4160
DS(      -   1005  1030  4091  9170  9290  9300
DUM      -   4092  4094  4096
FD       -   9245  9300
FLP      -   9245  9300
FM       -   9245  9300
FY       -   9245  9260  9270  9330
I        -    140   310  1030  2030  2070  3040  4050  4092  4094  4095  4096
             4100  4160  4189  4200  9270  9280  9290  9300
J        -   4092  4094  4095  4189
K        -   9510  9520
LD       -   9245  9300
LLP      -   9245  9290  9330
LM       -   9245  9290
LP       -   9150  9170
LP1      -   9242  9245

*** COMMONed variable list

*** Function list

*** Line number references
  280 -     310
  300 -     300
 2010 -    2030  2130
 2040 -    2020
 2050 -    2070
 2080 -    2060
 2120 -    2120

*** Subroutine references
 1000 -     140
 2000 -     310
 3000 -     310
 4000 -     310
 5000 -     310
 9000 -     160   220
 9100 -    2010  2050  3020  4030

*** Return to line number

*** File references
* File OPENed
* File LOADed
* File MERGEd
* File CHAINed

**************************************************************************************
Summary Statistics
   Number of program lines :        91
   Number of unresolved lines :      0
   Number of variables :            41
   Number of functions :             0
   Number of subroutines :          10
   Number of files :                 0
```



PERSONAL CASH FLOW MANAGER - PART II

By Gary Oppenheimer

Special Requirements: Personal Cash Flow Manager - Part I
Files Used: CASH.BAS
CASH-RPT.BAS
CASH-BLD.BAS
ACCOUNT.CDE
LEDGER.CDE
TRANS.DAT
ARCHIVE.DAT

Part I of the PERSONAL CASH FLOW MANAGER *enabled you to maintain a complete and integrated record of all your checking account activity. You created a list of checking accounts and a list of charge categories (i.e. income and expense types) that were the two reference files of the system. They insured the integrity of every entry into the third file of the system, the transaction file. Every trans-*

action had to be associated with a checking account and a charge category. The interconnection of these three data files not only assured internal consistency, it also automatically provided running balances by account and charge category.

Procedures to maintain these files—to add, change, delete and peruse the information in them—were an essential feature of the PERSONAL CASH FLOW MANAGER *Part I. Also at your disposal were a number of ways to display or print your checking account activity. Furthermore you could reconcile your computer records with your monthly bank statement, and subsequently clear these reconciled transactions to begin recording activity for a new month.*

PART II IS A COMPLETE CASH TRACKING SYSTEM

Part II of the PERSONAL CASH FLOW MANAGER *incorporates all the capabilities of Part I, and adds several new features which make this a truly comprehensive personal cash management system.*

First, Part II introduces the ability to make inter-account transfers. This small change enables you to add all your other cash accounts to the system, such as savings and credit cards.

Second, Part II allows you to set a budget each month for your income and expenses by charge category. New displays and reports have been added to let you compare your actual cash activity to budget.

Third, the PERSONAL CASH FLOW MANAGER *will now maintain a running 24 month history of your cash activity summarized by charge category. This facility sophisticates your month-end processing, and also makes a historical record of your cash activity available for analysis.*

CREATION

In order to use the *PERSONAL CASH FLOW MANAGER* Part II you must first build the new version of the program yourself. The programs you have received on your *PC Disk Magazine* diskette as Part II will not work by themselves. They must be combined with a key element from your prior version of the *PERSONAL CASH FLOW MANAGER*, Version 1.0. The procedure to accomplish this integration is as follows.

Begin by making a working copy of the new programs comprising Part II of the *Personal Cash Flow Manager*. Put your *PC Disk Magazine* diskette in the default drive and type:

```
COPY CASH*.* B: ↵
```

where B: is the drive containing a blank, formatted work diskette. Three files, CASH.BAS, CASH-RPT.BAS and CASH-BLD.BAS, should be copied over.

Now load Advanced BASIC into your PC by typing:

```
BASICA ↵
```

The next step is to extract the needed program section from your earlier version of the system. You should have a working diskette with Version 1.0 of the *PERSONAL CASH FLOW MANAGER*, created from the first issue of *PC Disk Magazine*. Put this diskette in the default drive and type:

```
LOAD"CASH ↵
```

When the CASH program has loaded type:

```
DELETE 1-14080 ↵
DELETE 50040-65000 ↵
```

Then replace the old working diskette of the *PERSONAL CASH FLOW MANAGER* with your new working diskette and type:

```
SAVE"TEMP",A ↵
```

You have now saved the portion of Part I needed to bring Part II to life. To complete the operation, load the new version of the CASH program into BASIC by typing:

```
LOAD"CASH ↵
```

and then integrate the critical Part I section with the command:

```
MERGE"TEMP ↵
```

Finally save the resulting workable program by typing:

```
SAVE"CASH ↵
```

The last step is to replace your earlier Version 1.0 of the *PERSONAL CASH FLOW MANAGER* with this new Version 2.0. Do so by copying the new programs over the old ones. With the new programs in your default drive (as they are at this point), type:

```
SYSTEM ↵
```

to exit Advanced BASIC, then:

```
COPY CASH*.* B: ↵
```

where B: is the drive which contains the old working diskette of the *PERSONAL CASH FLOW MANAGER*. When the copy operation completes, your old Version 1.0 working diskette is now your new Version 2.0 working diskette.

(Note: We are aware that there was a problem running Version 1.0 of the *PERSONAL CASH FLOW MANAGER* under DOS 1.1. Simple instructions to correct the problem were provided to all readers requesting them. Needless to say this problem has been corrected in the new, Version 2.0 of the system. The program section taken from Version 1.0 will not recreate the problem).

YOU MUST BUILD THE NEW VERSION OF THE PROGRAM

START-UP

The new version of the *PERSONAL CASH FLOW MANAGER* introduces a new, fourth data file, the history file. The addition of this fourth file requires that you run Advanced BASIC with additional options in effect. Specifically, every time you want to use the new version of the *PERSONAL CASH FLOW MANAGER*, you should load Advanced BASIC into your PC by typing:

```
BASICA/F:4/S:512 ↵
```

Then put your updated working diskette in the default drive and type:

```
RUN"CASH ↵
```

As with the prior version of the system, the program begins with the title screen.

The first time you run your new version, one of two things will happen after the title screen. If you already have an account, charge category and transaction file from Version 1.0, the program will automatically convert them to the new formats used by Version 2.0. All the existing contents will be preserved. If these files do not already exist, the program will create them for you in the new format, initializing them as empty.

PART II ALLOWS INTER-ACCOUNT TRANSFERS

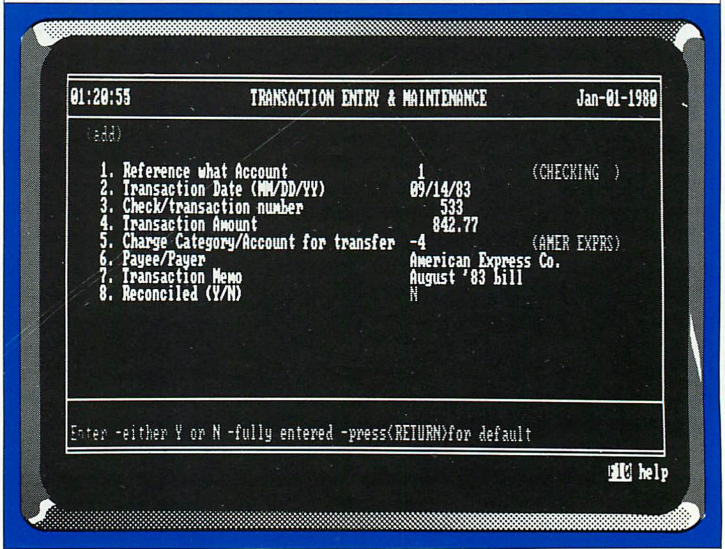
Once your data files are in the new format, the main menu will appear. As with Version 1.0, future uses of the program will take you directly from the title screen to the main menu.

You will see that the main menu looks the same as it did in Version 1.0. In point of fact Version 2.0 operates in the same manner as its predecessor, with the exception of several key additional facilities described below. To simplify and streamline the documentation here, we assume that the reader is familiar with all the basic procedures of Version 1.0. This article will document in-depth the new features of the Personal Cash Flow Manager—Part II.

INTER-ACCOUNT TRANSACTIONS

In Version 1.0 of the *PERSONAL CASH FLOW MANAGER*, you could credit money to an account or debit money from an account, but in either case the exchange of funds was always between your cash reserves on the one hand and the outside world on the other. There was no way to move money around within your own personal financial system, say from your checking account to a savings account.

As a result, the first version of the *PERSONAL CASH FLOW MANAGER* was really limited to the maintenance of checking account information, since checking accounts most closely mirror this "money in, money out" relationship to the outside world. You might have



```
01:20:55          TRANSACTION ENTRY & MAINTENANCE          Jan-01-1980

(add)

 1. Reference what Account              1              (CHECKING )
 2. Transaction Date (MM/DD/YY)         09/14/83
 3. Check/transaction number               533
 4. Transaction Amount                  842.77
 5. Charge Category/Account for transfer -4            (AMER EXPRS)
 6. Payee/Payer                         American Express Co.
 7. Transaction Memo                    August '83 bill
 8. Reconciled (Y/N)                    N

Enter -either Y or N -fully entered -press(RETURN)for default
                                                              F10 help
```

An inter-account transfer—paying the American Express bill.

handled the transfer of funds to other cash accounts by creating two generic and complementary charge categories for this purpose, (a "Transfer Out" debit category and a "Transfer In" credit category). This arrangement would work, but it necessarily limited the scope and thereby the completeness of the cash activity you could record.

Version 2.0 lifts this limit by introducing inter-account transactions. Now you can record your transaction activity for all your cash accounts, up to the system limit of nine accounts in all. Create accounts for your credit cards for example. Enter all your credit card charges as transactions. Then, when you pay your credit card bill each month, simply enter an inter-account transaction to debit the appropriate internal source of funds account, and credit the particular credit card account. Do the same with your record of savings account activity. In short any account you maintain that can be the origin or destination of cash can now be tracked by the *PERSONAL CASH FLOW MANAGER*. Note however, that you may still wish to treat relatively inactive accounts, such as an IRA, as a charge category rather than an account. From a cash flow standpoint such an account could be considered essentially an expense (or if you're collecting benefits, an income) category. The choice is yours.

You can create your new accounts as you did in Version 1.0, beginning with the selection of choice 2 from the main menu.

Entering an inter-account transaction is simple. Go into transaction maintenance mode by selecting choice 3 from the main menu. Once there, press:

F2

to add a transaction. For the account number, enter the number of the account which is the source of funds, that is, the account to be debited. Enter the date and amount as usual. Then for charge category enter the number of the account to receive the funds, preceded by a hyphen. It is this hyphen in the charge category field which identifies the transaction as an inter-account transfer. The system will display the name of the account referenced, just as it would the charge category in a regular transaction. When the inter-account transaction is complete the system will automatically adjust the balances for both accounts by the amount of the transaction.

For display and reporting purposes these inter-account transactions are treated like ordinary transactions. The hyphen appearing in the charge category field, which precedes the name of the credited account, is what identifies them as different. They appear along with all other transactions, in chronological order within account (the debited account for these transactions), for both the Integrated File Display and the transaction reports. For the transaction reports, inter-account transactions are obviously not reflected in the charge category summary, but only in the account summary.

BUDGET YOUR CHARGE CATEGORIES ON A MONTH-BY-MONTH BASIS

BUDGETING

This new version of the *PERSONAL CASH FLOW MANAGER* introduces the ability to enter a budget for your charge categories on a month-by-month basis as part of your normal charge category maintenance. You can enter a budget amount for the current month only, for any and all charge categories. When you archive your current month's data, discussed in the next section, the budget amount for each category for the current month will be archived as well.

This makes budget information available for historical review. At the same time the archiving process will carry last month's budget forward as the new month's budget, for each charge category in the system. Therefore you need only enter the budget amount once for each charge category, rather than every month. Of course you can change the current month's budget at any time if you choose through charge category maintenance. Budget information is reflected in the Integrated File Display, as well as a new report.

In order to set up a monthly budget, the first thing to do is to select the Charge Categories Maintenance function from the Main Menu. So, push:

1

ALL BUDGET AMOUNTS ARE INITIALLY SET TO ZERO

A Charge Category Maintenance screen will appear, listing any charge categories you have already entered. Initially all budget amounts in the system are zero, as set by the routine which creates the new file formats. To add or change a monthly budget entry to any of your existing charge categories, just push:

F3

to alter a charge category. Following this entry, you'll be asked to indicate the number of the charge category you want to alter. Enter the appropriate number. ''Enter function for inquiry'' will appear next on the screen. Again, push:

F3

to alter the charge category record currently displayed. ''Enter field to alter or return to proceed'' then appears. Now, press:

4

to select the budget field for alteration. When the cursor flashes at the budget line, enter the appropriate budget amount, rounded to the nearest dollar, followed by the Enter Key.

You can now use the F1 and F2 function keys to move through all your charge categories, following the procedure just described to enter budget amounts for each. When you're done, exit the alteration function as you would normally by pressing:

Esc

That's all there is to entering and changing your current month's budget.

The format of the Integrated File Display has been changed slightly to accommodate this new budget information. In the charge categories window you will see that there are now two columns of numbers. The first is the budget amount for the charge category, the second is the actual month-to-date charge category total. The cents have been dropped from the actual amounts in order to fit the information.

If the actual amount for the charge category exceeds the budgeted amount, the information for that category will be displayed in reverse video, to highlight it for you. As mentioned above, all budget amounts will initially be set to zero. Consequently at first any category with a non-zero actual amount will be highlighted. This misleading state of affairs should motivate you to enter budget amounts, and thereby have only true over-budget categories highlighted.

To make the budget information really useful to you, a report has been added to the Report/Inquiry submenu. (You'll recall that you get to this submenu by selecting choice 5 from the main menu). You will see that this submenu has a new choice, number 7, to "Display Archived Budget Information." When you make this selection, a screen will appear asking you, at the bottom, to specify the number of the charge category which you would like to have reported.

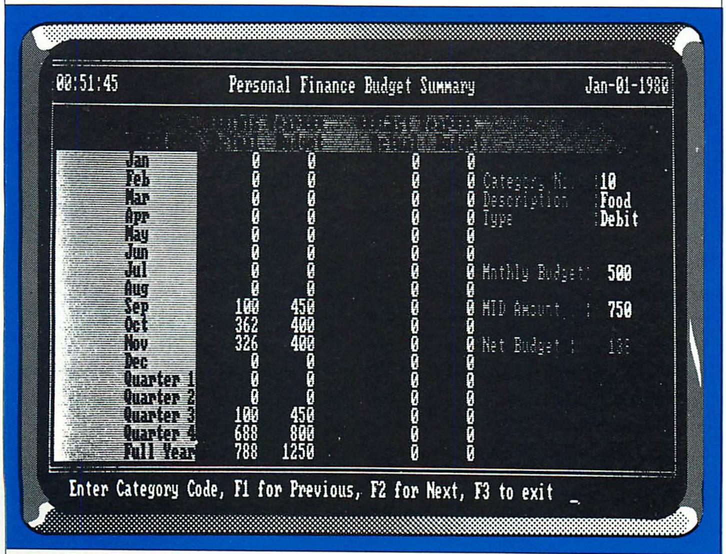
When you enter a valid category number the computer will display all historical budget and actual charge category figures for the current year and last year by month, with quarterly and year-to-date summaries. On the right hand side of this display is various charge category identifying information, along with budget and actual amounts for the current month. Lastly, and quite importantly, below the current month's numbers you will see a "Current Month Net Budget." This number represents the difference between the total budget for credit categories and the total budget for debit categories in the current month. A positive number indicates that you have budgeted more income than expenditures in the current month. A negative number indicates that you expect to be in the red this month.

A REPORT HAS BEEN ADDED TO DISPLAY BUDGET DATA

Unlike the other reports, this budget report does not have a separate print selection to generate hardcopy. To get a printed copy of the budget report for a given charge category, use the IBM Print Screen Facility, i.e. with a complete report on your screen, press:

⇧ PrtSc *

Once a valid charge category has been reported, you can quickly display the budget report for the following or prior budget category



```
00:51:45          Personal Finance Budget Summary          Jan-01-1980

         [illegible]
         Jan      0     0          0     0
         Feb      0     0          0     0 Category No.   :10
         Mar      0     0          0     0 Description    :Food
         Apr      0     0          0     0 Type           :Debit
         May      0     0          0     0
         Jun      0     0          0     0
         Jul      0     0          0     0 Mnthly Budget:  500
         Aug      0     0          0     0
         Sep    100   450          0     0 MTD Amount   :  750
         Oct    362   400          0     0
         Nov    326   400          0     0 Net Budget :   [illegible]
         Dec      0     0          0     0
   Quarter 1      0     0          0     0
   Quarter 2      0     0          0     0
   Quarter 3    100   450          0     0
   Quarter 4    688   800          0     0
   Full Year    788  1250          0     0

 Enter Category Code, F1 for Previous, F2 for Next, F3 to exit _
```

The new budget report.

by using the function keys F1 and F2, as indicated at the bottom of your report screen. To end this report mode and return to the Report/Inquiry submenu, press:

F3

Note: On the Report/Inquiry submenu you will also see another new option, number 8, called "Output Data for Spreadsheet Software." This facility, which will allow you to transfer your historical charge category information to one of four popular spreadsheet packages, will be implemented in Part III of the *PERSONAL CASH FLOW MANAGER*. If you select it at present, you will be told that it is not yet installed.

ARCHIVING

The ability to record a monthly history of your cash activity is one of the most important new features introduced with Part II of the *PERSONAL CASH FLOW MANAGER*. You will recall that in Part I there was no facility to save data from past months, except to let it accumulate in your transaction file (a bad idea given the limited transaction space available). After reconciling the transactions for a month, your only option was to delete those transactions from the system.

Version 2.0 offers a way to save the reconciled data from last month, by providing a new facility to close the books for a month. The system will now maintain a running 24-month history of your cash activity, month by month, in the file ARCHIVE.DAT. With every new month added, the oldest month gets dropped from the file.

PART II OFFERS A NEW FACILITY TO CLOSE THE BOOKS FOR A MONTH

Activity number 4 on the main menu is now "Transaction Reconciliation & Month-End Closing". When you select number 4, you are first asked if you want to perform reconciliation or closing. Enter:

R

to undertake transaction reconciliation, or:

C

for month-end closing. Pressing:

Esc

will return you immediately to the main menu.

Transaction reconciliation proceeds just as it did in Version 1.0 of the system. However, when you finish reconciling you are not asked about clearing them out, you are simply returned to the main menu.

Before you begin the month-end closing activity, be sure to make a backup copy of all your data! The month-end closing routine makes permanent changes to your data, so a backup is critical just in case there's a problem. Make your backup by putting a blank working diskette in a second drive and typing:

```
COPY *.DAT B: ↵
COPY *.CDE B: ↵
```

where B: is the letter identifier of that second drive. (If you don't have a second drive, type this command anyway. The computer will let you switch diskettes in the one drive as part of the copy operation). The first command should copy two files, TRANS.DAT and ARCHIVE.DAT. The second command should copy another two files, LEDGER.CDE and ACCOUNT.CDE. If you had to restore these files you could simply copy them back to your Version 2.0 working diskette.

The purpose of month-end closing is to create an accurate summary by charge category of your cash activity in the month just past. The individual transactions for the month are not themselves saved, because that would take too long and provide more detail than is necessary (that's what you have paper records for). Rather, the individual transactions are aggregated into charge category totals for the month, and those totals are then saved along with their associated budget amounts for the month. The individual transactions are then deleted from the system.

The key to creating accurate historical information is to make sure that all transactions for the month have been reconciled with the appropriate institutional statements. Such reconciliation provides the external check on the information you have entered into the computer. Thus before you close the books for a given month, you should reconcile all the transactions for that month.

CHARGE CATEGORY TOTALS GET ARCHIVED

When you choose to perform month-end closing, the computer will ask for the date of the last day of the month you wish to close. When you enter a valid date, the computer will scan all the transactions currently in your file to see if there are any transactions in the month to be closed which have not been reconciled. Every such transaction found (if any) will be presented to you for confirmation. You must confirm that this is a valid transaction and is intentionally unreconciled. This process insures that no transactions have been overlooked. Since no data has been changed or archived at this point, you can press:

```
Esc
```

during this process to abort the close and make any needed corrections.

If there are no exception transactions, or if you confirm as valid all those that are presented, the computer will then ask you to confirm that closing should commence. This is your last chance to bail out. If you proceed, the program will: 1) move every month of data in the history file back one place, dropping the 24th month ago and opening a space for the new month's data, 2) compute the charge category totals for the month being closed from the reconciled transactions and then write those totals and their associated budget amounts to the history file, 3) subtract the reconciled transaction amounts from their associated current charge categories, so that the totals for the month in progress (the month after the one being archived) are properly adjusted, and 4) delete the reconciled transactions from the system.

You can see why we emphasize making a backup copy, considering the extensive changes wrought by month-end closing. A few additional comments concerning month-end closing are relevant.

First, be sure to close one month before reconciling any transactions for the following month. Otherwise reconciled transactions from one month and the next will get combined in the archived category totals for the earlier month.

Second, once a month has been closed, it cannot be changed by the system. It is effectively carved in stone. You can restore your backup files, make changes, and then archive the same month again if you must make corrections.

FREE AND USED SPACE ARE DISPLAYED IN FILE UPDATE MODE

Third, you should always close months in consecutive order. Never skip a month. Skipping a month will throw off your budget report, since the calendar in this report assumes that your historical data is in consecutive month order. If you want or need to skip a month, then perform a dummy closing for the month, that is, a closing with no reconciled transactions to process.

Fourth and finally, be aware that the system will often have a combination of last month and current month transactions present simultaneously. On the one hand it is necessary to retain last month's transactions into the current month in order to reconcile them when statements are received. On the other hand you can't very well wait until all of last month's transactions are reconciled before entering any of this month's transactions.

As a result, your charge category month-to-date amounts often reflect the combination of two month's data. They can therefore be misleading when compared to a budget amount intended to be one month's total. The charge categories will accurately reflect the current month's actual amount after you close the prior month, up until you start entering transactions for the following month.

SYSTEM LIMITS

The first issue was not sufficiently clear on this important topic, so a very brief summary appears here for your reference.

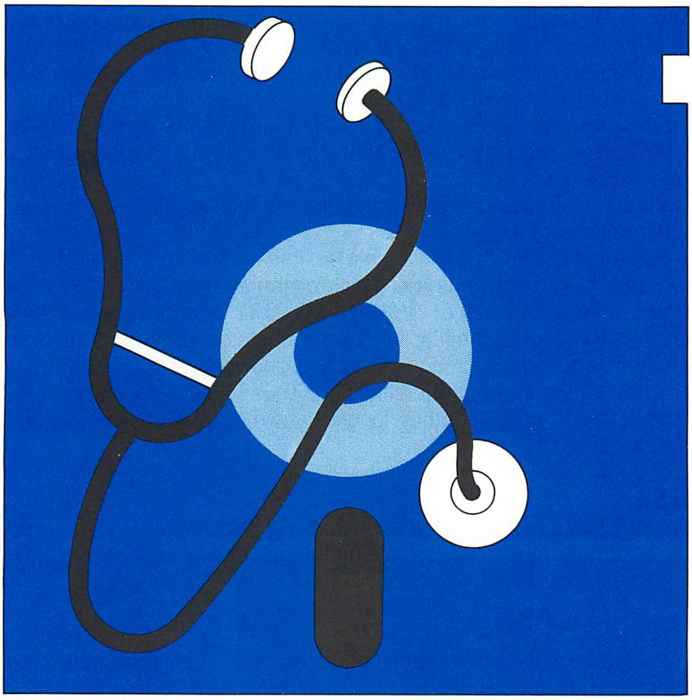
The maximum number of items that can be managed by the system are:

9	accounts.
99	charge categories.
250	transactions (in Version 2.0)
24	consecutive months of historical data.

One other limiting factor in your use of the *PERSONAL CASH FLOW MANAGER* is the amount of main memory you have in your computer, which in turn determines the amount of memory available to you in BASIC. Any PC with at least 128K of main memory will allow the maximum BASIC work space of about 60K. This gives you the full capacity of the *PERSONAL CASH FLOW MANAGER*.

If you have a PC with less than 128K of main memory, you should run the program under BASIC rather than Advanced BASIC, since the former requires less of your limited memory for itself. When you load BASIC or Advanced BASIC, the sign-on message tells you how many bytes are free. If this number is less than 50K, you may not be able to reach the capacity ceilings listed above.

UTILITIES & DIAGNOSTICS



DISK LABEL
By Peter Norton

Special Requirements: None
Files Used: DLABEL.COM

There you are in your den or office with diskettes stacked everywhere. Some are labeled; some are not. Some of the labels are legible; others not. Even if you maintain your diskettes meticulously, mishaps, which can render your diskette unidentifiable, are unavoidable.

You need protection from the dangers of bad penmanship and casual housekeeping. DISK LABEL, *by Peter Norton, offers comprehensive and reliable control of your diskettes through the use of software volume ID labels. These labels are actually part of the diskette contents, and therefore cannot be separated from the* *diskette or physically damaged without damage to the diskette itself.* DISK LABEL *is another element from the widely praised Norton Utilities for the IBM PC. It is easy to use, and helps to efficiently manage your growing diskette library.*

WITH *DISK LABEL* YOU CAN LABEL ALL YOUR DISKETTES

BACKGROUND

Disk volume ID labels were introduced with DOS 2.0. However, DOS does not fully support them. For example, you can only place a label on a diskette when you format it, and even then, only if you use the new nine-sector format of DOS 2.0. There is no means to add a label to an existing diskette or to change a label. Furthermore, there is no way to label DOS 1.1's standard eight-sector diskettes. *DISK LABEL* solves these problems by giving you complete control over diskette volume ID labels. With *DISK LABEL* you can systematically label all your diskettes, and change this label when the diskette contents change.

Because *DISK LABEL* is such a flexible and reliable identification tool, we have used it as the basis for the *DISK INVENTORY* program, also in this issue. *DISK INVENTORY* is a complete library system for your diskettes. In order to catalog all your diskettes, each one must have a unique identifier. The volume ID label was the natural choice for such identification. Thus *DISK LABEL* and *DISK INVENTORY* work together to give you comprehensive diskette management.

USE

To use *DISK LABEL*, just type the name of the program file followed by the letter of the drive containing the diskette to be labeled.

For example:

```
DLABEL A ↵
```

If you leave off the drive specification, *DISK LABEL* will default to the B drive. Thus, the simplest way to use *DISK LABEL* is to put your *PC Disk Magazine* diskette in your A drive and the diskette to be labeled in your B drive. Then just type:

```
DLABEL ↵
```

The program first displays a two-line identifying banner and then tells you the current volume ID label of the diskette being processed. If there is no current volume ID label, you have the option to enter a label up to eleven characters long, or simply press:

↵

to leave the label blank. For a diskette that currently has a volume ID label, you can type in a new label, or press:

↵

to leave the current label unchanged, or press:

Del

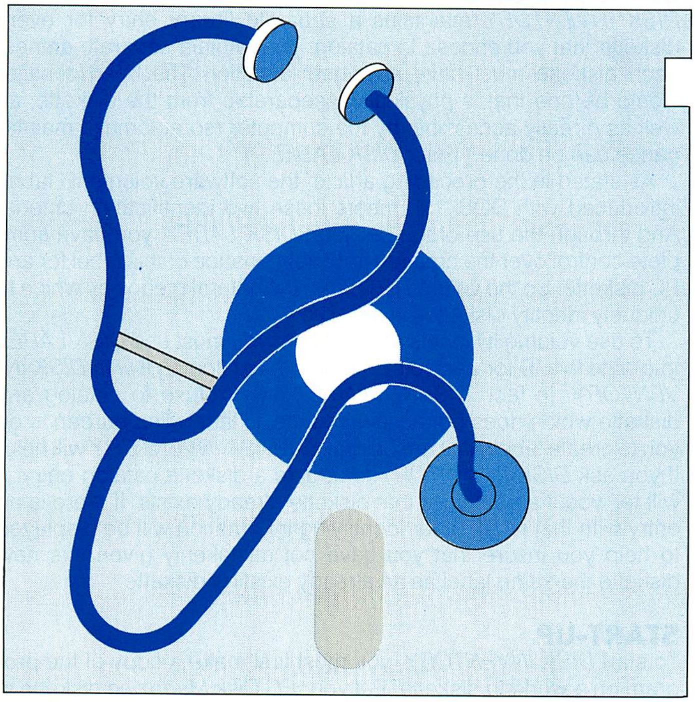
to delete the current label, leaving it blank.

When you have entered the label you wish, press:

↵

to tell *DISK LABEL* to write the new label onto the diskette.

UTILITIES AND DIAGNOSTICS



DISK INVENTORY

By Mark Zachmann

Special Requirements: None
Files Used: DISKINV.EXE
INVDISK
INVFILE

We all know that a computer can help better organize both personal and business information. However, in the process you still have to manage the diskettes which contain the information. Ever wonder which diskette contains a specific file you need? Or whether the file currently in hand is the latest version? If so, then you know that the computer creates a whole new set of information management problems.

Before you throw out your diskettes in disgust, try using DISK INVENTORY, *a comprehensive library system for all your diskette information.* DISK INVENTORY *lets you maintain a catalog of all your diskettes and their file contents. You can add, change or delete* *diskette entries automatically. Once you've created your diskette catalog, use* DISK INVENTORY *to summarize your diskette entries, to display a specific entry, or to find a particular file or group of files across all your library contents. With* DISK INVENTORY *you can use the computer to help manage its own information explosion.*

BACKGROUND

DISK INVENTORY maintains a separate library entry for every diskette that you choose to catalog. To maintain separate entries, each diskette must have a unique identifier. The best identifier would be one that is physically inseparable from the diskette, as well as directly accessible by the computer (so automatic maintenance can be done.) Enter *DISK LABEL*.

YOU MUST LABEL YOUR DISKETTES USING *DISK LABEL*

As stated in the preceding article, the software volume ID label, introduced with DOS 2.0, meets these two identification criteria. And through the use of the program *DISK LABEL*, you have complete control over the creation and maintenance of this label for any PC diskette. So the volume ID label is the natural means by which to uniquely identify diskettes in the library.

To use volume ID labels as identifiers you must use *DISK LABEL* to create this ID for each diskette before cataloging it with *DISK INVENTORY*. In fact, *DISK INVENTORY* will refuse to catalog any diskette which does not have a volume ID label. The burden is on you to create labels that are unique, but *DISK INVENTORY* will help. If you ask *DISK INVENTORY* to update a diskette catalog entry, it will tell you if an entry for that diskette already exists. If there is an entry with that label, other identifying information will be displayed to help you insure that you have not mistakenly given this new diskette the same label as an already existing diskette.

START-UP

To start *DISK INVENTORY*, you must first make a copy of the program on a working diskette. Put your *PC Disk Magazine* diskette in the default drive and type:

```
COPY   DISKINV.EXE   B: ↵
```

where B: is the drive containing the blank working diskette. Then put that working diskette in the default drive and type:

```
DISKINV ↵
```

If you are running a black and white monitor from the color/graphics adapter card, you should invoke the program by entering this abbreviated program name followed by any single letter, for example:

```
DISKINV B ↵
```

This additional parameter causes the program to modify its screen output to suit a black and white monitor.

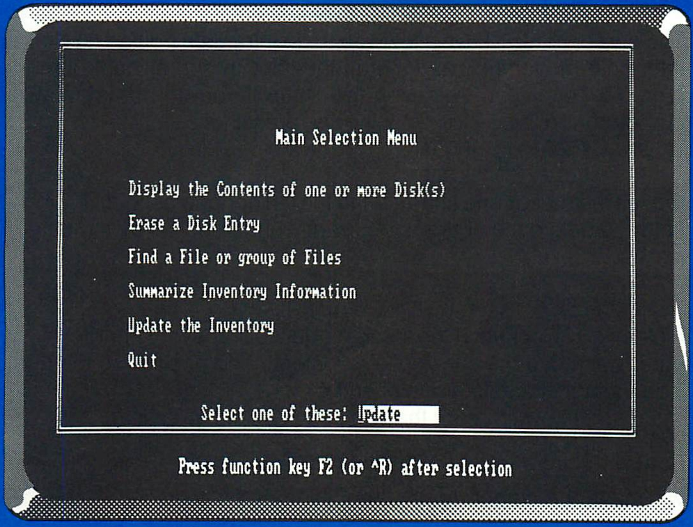
The first thing you will see is the title screen. If you are running the program for the first time you will see a message on the title screen stating that your library files, INVDISK and INVFILE, are being created. To begin execution of the system, press:

```
↵
```

as stated on the title screen.

GENERAL PROCEDURES

After the title screen, you are presented with the Main Selection Menu for the *DISK INVENTORY* system. On this main menu are listed all the activities available in the system. The activities fall into two basic categories: maintenance and retrieval. Before describing each library activity, a few universal operating procedures will be explained.



```
Main Selection Menu

Display the Contents of one or more Disk(s)
Erase a Disk Entry
Find a File or group of Files
Summarize Inventory Information
Update the Inventory
Quit

Select one of these: Update

Press function key F2 (or ^R) after selection
```

The Main Selection Menu—ready to update the catalog.

When *DISK INVENTORY* offers you a specific list of choices, such as on the Main Selection Menu, you make your choice by entering only the first letter of the associated choice phrase. You do not have to press the Enter key to register your selection. When the letter is typed, the associated word or phrase will appear in the appropriate choice location. For example, if you want to erase a diskette entry, press:

E

at the Main Selection Menu, and this choice will appear at the bottom of the screen.

When you make a choice in this manner, that selection is not immediately executed. In *DISK INVENTORY* the specification of a choice and the execution of that choice are two different actions. To specify a choice, you will commonly fill in one or more fields on a screen. You can keep changing your screen specification as much as you like, with no resulting action, until the screen reflects what you want. To cause the choice displayed on the screen to be executed, you must press:

F2 or Ctrl R

Alternatively, to cancel the activity specification you are in, you must press:

F10 or Ctrl X

MAKE YOUR CHOICE BY ENTERING ONLY THE FIRST LETTER

Only by pressing one or another of these keys can you cause *DISK INVENTORY* to do anything. The cursor can be located anywhere on the screen at the time you press these action keys. In addition to functioning as "execute" and "cancel" keys, in circumstances requiring a yes or no answer these keys will operate as "yes" and "no" respectively.

You can delete the current entry in any choice field when the cursor is on that field by pressing:

Esc

The cursor control keys allow you to do intra-field editing as well as inter-field movement, as long as they are in cursor control mode (see the Technical Preface in this issue if you do not know what cursor control mode is). Specifically, pressing:

4 ← or 6 →

moves the cursor one character to the left or right respectively in the field. To delete the character currently underscored by the cursor, press:

* Del

Field entry is locked in the insert mode, so that every character you enter will move the current field contents one position to the right. You cannot alter this mode of entry.

On screens with more than one choice field, pressing:

8 ↑

will move the cursor to the beginning of the previous field, while pressing:

2 ↓ or ↵

will move the cursor to the beginning of the next field.

As a matter of general procedure, you should note that *DISK INVENTORY* supports the standard wildcard conventions for file names (i.e., the use of ? and * characters). *DISK INVENTORY* also allows you to use these same wildcard conventions when specifying disk labels.

Finally, you can control the printing of screen output by using the standard DOS display toggle:

Ctrl S

If you want to cancel a listing process that is in progress, press:

Ctrl Scroll Lock

and you will then be able to press the Enter key to return to the previous choice screen, as you would under normal termination.

YOU CAN ADD, MODIFY AND DELETE DISKETTE ENTRIES IN YOUR LIBRARY

DISKETTE MAINTENANCE

The Main Selection Menu activities "Update the Inventory" and "Erase a Disk Entry" enable you to add, modify and delete diskette entries in your library. The update activity includes the add and modify functions. To add or modify a diskette entry you must put the diskette in the disk drive, and let the computer determine which to perform (i.e., the computer determines whether an entry with this label already exists). If it is your intention to add a diskette, you should fill in the "Printed Label" and "Description" fields before invoking execution of the update activity with the F2 function key. If you don't, these fields will initially be set to blank. If you want to change these fields for an existing entry, type what you want in them, and your new entries will override the current values.

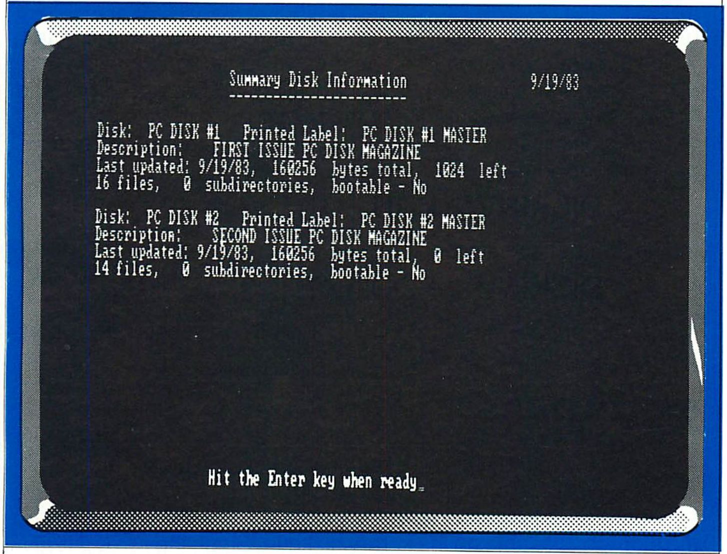
To erase a catalog entry simply enter a label that is on file. You may not use the wildcard characters here.

DATA RETRIEVAL

DISK INVENTORY provides three functions for the retrieval and display of your diskette library contents. If you want to know all the files resident on one or more diskettes, you should select "Display the Contents of One or More Disk(s)" from the Main Selection Menu. You then have the option to specify a particular disk by label, or a group of disks by using wildcard characters, or all disks by entering just an asterisk (this last case would give a complete list of all files in the library). In addition, you may request that the files be listed in a sorted order within disk, by up to two sort fields. You may choose to sort on any one of four file characteristics: Name, Extension, Date of last update, or Size, by entering the first letter of the characteristic. If more than one disk is selected, the disks will be listed in the order in which they were entered into the system.

YOU CAN GET A LIST OF ALL FILES THAT MEET UP TO THREE CRITERIA YOU SPECIFY

Suppose, however, that you would like to know just the characteristics of one or more diskettes in your library, without the details of the files they contain. You can choose "Summarize Inventory In-



```
              Summary Disk Information              9/19/83
              ------------------------

Disk:  PC DISK #1   Printed Label:  PC DISK #1 MASTER
Description:    FIRST ISSUE PC DISK MAGAZINE
Last updated: 9/19/83,  160256  bytes total,  1024  left
16 files,   0  subdirectories,  bootable - No

Disk:  PC DISK #2   Printed Label:  PC DISK #2 MASTER
Description:    SECOND ISSUE PC DISK MAGAZINE
Last updated: 9/19/83,  160256  bytes total,  0  left
14 files,   0  subdirectories,  bootable - No




              Hit the Enter key when ready_
```

Output of Inventory Summary—two issues so far.

formation" from the Main Selection Menu, and then specify the label of the disk(s), (in the same form as above) that you would like to see summarized. Once again, the disks will be displayed in the order in which they were entered into the system.

Lastly, and of great use in managing your files, is the option to "Find a File or Group of Files". With this activity you can get a list of all files in your library that meet up to three criteria you specify. Those criteria can be any combination of values for the file characteristics Name, Extension, Date of last update, and Size. To use less than three criteria, blank out just the field containing the name of the file characteristic by pressing:

Esc

The criteria that you do specify are taken as combined requirements, that is, to be selected a file must satisfy criterion one and criterion two and criterion three. The files that satisy all criteria will be listed in the order in which the diskettes containing them were entered into the system.

This facility can be used to find all copies of a particular file, all files of a particular type (by Name and/or Extension, including the use of wildcard characters), all files that haven't been touched since a certain date, etc., etc. This facility alone makes *DISK INVENTORY* an invaluable file management tool.

OUTPUT

By default all output from *DISK INVENTORY* goes to your screen display. Thus one way to obtain hard copy is to use the IBM PC Print Screen Facility when information of interest is displayed. This is fine for single screen displays, but what about longer listings?

YOU CAN SPECIFY THAT ALL LISTING OUTPUT IS TO GO TO A DISK FILE OR DIRECTLY TO THE PRINTER

DISK INVENTORY also allows you to take advantage of a feature called Redirected I/O. This means that when invoking the program, you can specify that all listing output is to go to a disk file or directly to the printer. Activity selection and choice specification still take place interactively on the screen, but when you invoke a function which generates a listing (e.g., Summary Disk Information, Diskette File Contents or File(s) Found), that listing will not appear on the screen, but goes to a disk file or the printer.

To exploit this feature of *DISK INVENTORY*, you must invoke the program with either the command:

```
DISKINV >PRN: ↵
```

to send output directly to the printer, or:

```
DISKINV >FOO.TXT: ↵
```

to send output to a file called FOO.TXT. You can subsequently sort, print, or edit the file FOO.TXT.

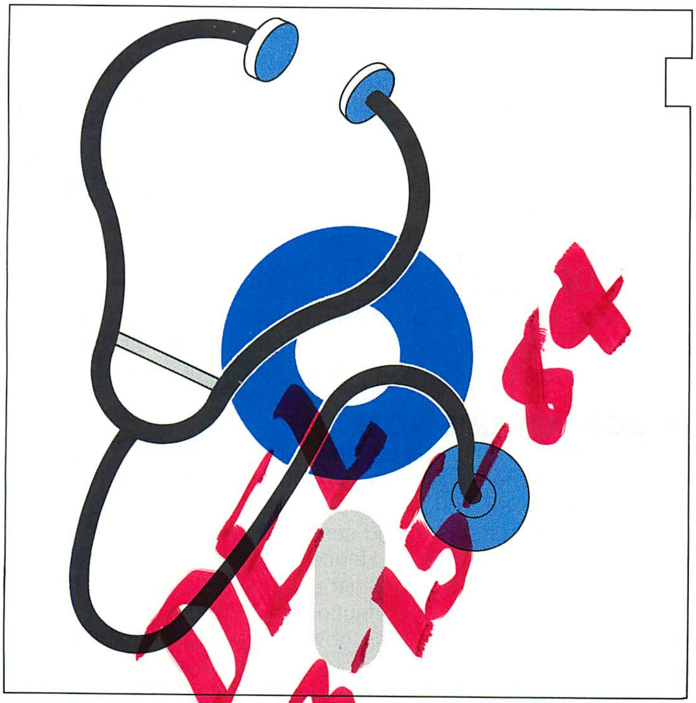
SYSTEM LIMITS

The diskette library maintained by *DISK INVENTORY* has no built-in limit as to the number of diskettes or files that can be accommodated. However, since the INVDISK and INVFILE reference files cannot span more than one diskette, the library size is constrained by the storage available on the library diskette. Since the *DISK INVENTORY* program must reside on the same diskette as the library itself, library file capacity must be computed as follows:

((Disk Capacity — 30K) / 2) / 32

Subtract 30K for the program from the total diskette capacity, and divide the result by 2 because the program makes backup copies of the INVDISK and INVFILE files every time it updates them. Divide this result by the 32 bytes/file the system requires (since files greatly outnumber diskettes, the latter need not be accounted for separately).

One closing note: *DISK INVENTORY* cannot catalog the diskette it is on. But, after all, that is now the only one you need to worry about.



DEV
3/15/84

DOUBLE VISION

By Bert Tyler

Special Requirements: Color/Graphics Adapter
Files Used: DOUBLE.EXE

Sometimes it's the little things that are really annoying. You want the most from your PC, and you're willing to spend the time and money to get it. So you make arrangements to hook up both a monochrome and color monitor to your system for the greatest display flexibility. Then, when everything is in place, you discover that you can direct screen output to either the monochrome or the color monitor, but not both. True, this may not be cause for homicidal rage, but it is maddening. Don't get mad, get DOUBLE VISION.

DOUBLE VISION *is a utility that extends the PC's Disk Operating System by enabling it to send display output to both your mono-*

chrome and color displays simultaneously. It restores to you the full display flexibility latent in a two-monitor system.

ALL TEXT APPEARS ON BOTH MONITORS

USE

To use *DOUBLE VISION*, simply put your *PC Disk Magazine* diskette in the default drive and type:

DOUBLE ↵

The program will check to ensure that you have both the monochrome and color/graphics adapters installed on your PC. If you do, both screens will be cleared and a sign-on banner will appear on both screens, telling you that Double Vision is on.

That's it. From that point on any text sent to the system display will appear on both your monochrome and color displays. Of course, graphics output can still only be sent to the color display.

To turn off *DOUBLE VISION*, type:

DOUBLE OFF ↵

You may notice a slowdown in your system's response time when *DOUBLE VISION* is active. This occurs because the display signal is being sent out twice, to two monitors instead of just one.

PROGRAMMER'S NOTE

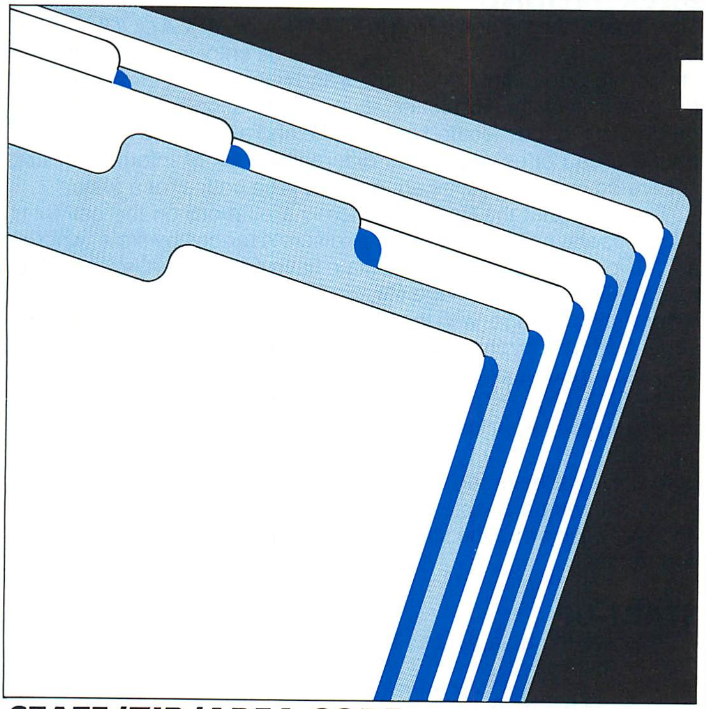
DOUBLE VISION works by intercepting the calls that DOS (and your programs) make to the ROM-based subroutines that control monitor output. *DOUBLE VISION* detects every such call made to your IBM monochrome monitor, and sends the very same message to your color-graphics monitor. Hence, *DOUBLE VISION* will work with any DOS-based program that uses the BIOS section of your operating system to display output on your monitor.

For certain types of programs, however, going through the operating system to send output to your monitor takes too much time. This is particularly true of spreadsheet programs and word processors where the amount of time it takes to update the screen is critical. Programs of this type tend to ignore the operating system completely and write directly to the monitor. With no subroutine calls to intercept, *DOUBLE VISION* does not work with these programs.

PROBLEM HANDLING

We try our best to thoroughly test all *PC Disk Magazine* software, and provide instructions that cover all aspects of its use. Nevertheless, error-free software and exhaustive documentation are elusive goals. So if you have a problem, please contact us and let us help. Although we hope you will not need it, the address to write to is:

PC Disk Magazine
Department 741
One Park Avenue
New York, NY 10016



STATE/ZIP/AREA CODE

By Morris Effron

Special Requirements: None
Files Used: STATEZIP.DAT
STATEZIP.BAS

You return to your home or office and find a message from someone you've never heard of before. The person has left a number for you to call with an area code of 808. Where in the world is area code 808, you wonder. What luck! Your latest copy of PC Disk Magazine *has just arrived, and it includes a program that will answer this question.*

Running STATE/ZIP/AREA CODE *to find the answer is probably not your first impulse. Quite frankly, it wouldn't be ours either. You would most likely call the operator and learn that area code 808 is*

Hawaii (a problem with your "Luau Lullabies" record order perhaps?) Nevertheless, this STATE/ZIP/AREA CODE *program puts at your disposal a complete, ready-to-use reference of such information, along with a data retrieval program that is both useful and educational.*

BACKGROUND

The name of the file containing the *STATE/ZIP/AREA CODE* table is STATEZIP.DAT. The file contains a one-line entry for each of the 50 states, plus the District of Columbia and Puerto Rico. Each state entry consists of the two-letter state abbreviation followed by the range of zip codes located in this state and a list of the state's telephone area codes. The three different pieces of information are separated by commas, as are multiple area codes for a state.

THE DATA FILE IS A STANDARD ASCII FILE

It appears that the Postal Service is a bit more on the ball than AT&T, insofar as zip codes fall into discrete ranges by state, whereas multiple area codes within a state have no apparent relationship to one another. Thus, in the file, zip code information for a state is recorded as a range, with the lowest and highest zip code numbers separated by a hyphen. The area codes follow in no particular order, separated by commas.

The file STATEZIP.DAT is a standard ASCII file, so that you can display it, print it, or call it into a text editor. However, the data is packed rather tightly and in this bulk form, is not very useful. Accordingly, we have included a BASIC program, STATEZIP.BAS, which will read the table into several BASIC variables and, if you choose, will look up the state location of any zip code or area code you request.

START-UP

To execute the data access program, load Advanced BASIC into your PC by typing:

```
BASICA ↵
```

Then put the *PC Disk Magazine* diskette in the default drive and type:

```
RUN "STATEZIP ↵
```

The screen will clear, and the program will tell you that it is "READING STATE/ZIP/AREA CODE DATA."

FILE DESIGN—A DIGRESSION

Reading the *STATE/ZIP/AREA CODE* data file is a little more difficult than you might think. The most direct approach would be to read the state abbreviations into one array, the zip codes into another, and the telephone area codes into a third. Doing so would make the data directly available for manipulation.

The problem with this direct approach is that each state does not have the same number of telephone area codes. Since some states have more than one, our program must have a loop to retrieve telephone area codes. But the number of area codes for any given state is unknown, so the question is: How many times should this loop be executed for each state? No constant number will work across the board, and we don't know in advance what variable number to use either. In short, we have no way to retrieve the correct number of area codes for each state.

We could have made the entry for each state the same length by using a random file structure for our data instead of a sequential file structure. (Random files by nature have fixed length records; see Appendix B of your IBM BASIC manual for more information on these contrasting file structures.) However, in a random file numeric data is commonly encoded, and the data records themselves usually have no separators between them. Hence a random file could not be displayed, edited, or printed like the sequential file we have provided. This makes it a less attractive alternative from the user's standpoint.

Rather than change the nature of our data file, we could have changed the contents of the file to give each state the same number of area codes. We could determine which state has the greatest number of area codes and then extend all the other state entries with dummy area codes such as zeros separated by commas, or even just commas. By appending the correct number of such dummy area codes to each state, all states would have an equal number of area codes.

The drawbacks of this approach are obvious. Aside from its inelegance (a serious fault to serious programmers), the resulting file would have a bewildering appearance and would use up more storage space. These same faults apply to the idea of adding a number to each state entry, indicating how many area codes the state has and then using that number to control the retrieval loop in our program.

THERE ARE ALWAYS COMPETING GOALS IN FILE DESIGN

The purpose of all this discussion is to illustrate the competing considerations that always exist when you design a data file. On the one hand, we would like a file that can be read directly into working variables. On the other hand, we want a file that uses space efficiently. Moreover, while economy of space is important, we would like to be able to display the file as is.

The result is a compromise, based on a weighing of priorities and an evaluation of trade-offs. Rarely is there a single "correct" design. Most often there are a number of defensible choices. The point here is to discover and thoroughly evaluate all your options.

Returning to the subject at hand, we had one other objective in choosing the file design for the *STATE/ZIP/AREA CODE* file. We wanted an example of a file you might typically create with a text editor in order to illustrate how you could read such a file from a BASIC program. In our last issue we offered one example of such a file, the *DEMOGRAPHIC DATA FILE*. Reading it was easy, because every entry had the same number of elements. The *STATE/ZIP/AREA CODE* file introduces a way to read text files with a variable number of elements per line.

DATA RETRIEVAL

The *STATE/ZIP/AREA CODE* program reads the data file a full line at a time, into the array RAW$ (line 105). Since we know there are 52 entries in the file, RAW$ is dimensioned at 52 elements. By retrieving an entire line at a time, we pick up whatever number of telephone area codes the state may have. The key to retrieving a full line of text in one shot, whatever its length, is the BASIC command LINE INPUT.

When all data file lines have been read, the program tells you so and asks if you want the raw data converted. At this point each element of RAW$ contains a line of the *STATE/ZIP/AREA CODE* data file exactly as it appears in that file. The data is not of much use in this form, since we can't manipulate the individual data elements. However, you may have other ideas about what you want to do with this raw data, so the program gives you the option to quit at this point. If you choose to quit, type:

N ↵

and you will be left in Advanced BASIC, with the array RAW$ at your disposal.

DATA CONVERSION

If you choose to continue, the program will proceed to convert the raw data lines into their usable constituents (lines 200, 270). Each element of RAW$ (i.e., each data line) is broken up, the pieces converted to numbers as necessary, and the results moved into the four arrays STATE$ (for the state abbreviation), LZIP (for the lowest zip code in the state), HZIP (for the highest zip code in the state), and ACODE% (for the telephone area codes in the state). STATE$, LZIP, and HZIP are all 52-element arrays. ACODE% is a two-dimensional array, 52 by 8, which allows as many as eight area codes (the maximum required) to be stored for each state.

OUR DESIGN REQUIRES US TO CONVERT THE DATA

This conversion processing is the price we pay for a file design that won't let the file contents be read directly into usable variables. The conversion only takes a few seconds, but this is because the file is small. If we were dealing with a considerably larger file, this additional processing time might be unacceptable. We could be forced to redesign our data file, sacrificing some of its current benefits, to reduce this conversion processing time. Such are the tradeoffs of file design.

When the conversion is complete, the program asks if you want to proceed to do any zip code/area code lookups. If not, you may end the program and return to Advanced BASIC by typing:

N ↵

When you exit at this point you have at your disposal the arrays STATE$, LZIP, HZIP and ACODE%, in addition to RAW$. If you want to continue, type:

Y ↵

The screen then clears, and the lookup menu appears.

LOOKUP

The zip code/area code lookup procedure (lines 330 to 1000) will tell you the state location of any valid zip code or area code you want to know. Using it is quite simple: First choose which of the two you want to look up; then enter the code to be looked up. That's it. The program will tell you what state that particular code is in or if there is no state with such a code. It will then loop back for another lookup selection. For example, if you want to look up area code 808, type:

2 ↵

in response to the CHOICE? prompt. Then enter:

808 ↵

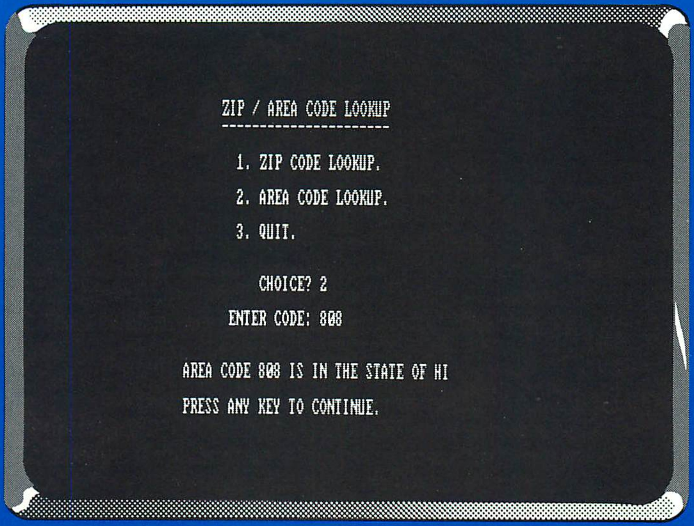
in response to the ENTER CODE: prompt. The computer will then tell you that area code 808 is in HI (Hawaii) and ask you to press any key to continue. When you do so, you will be back at the CHOICE? prompt. Enter:

3 ↵

to end the program.

A closing remark on programming trade-offs. If you examine the program section that does lookups you will see that a "brute force" approach has been used. The program simply looks sequentially through the relevant codes for every state, until it either finds the one sought or exhausts the entire list—a very inefficient way to do lookups. Far more sophisticated algorithms exist, but they require that your reference data be in some kind of order.

CHOOSE WHAT TO LOOK UP, THEN ENTER THE CODE



```
ZIP / AREA CODE LOOKUP
----------------------

 1. ZIP CODE LOOKUP.
 2. AREA CODE LOOKUP.
 3. QUIT.

    CHOICE? 2
 ENTER CODE: 808

AREA CODE 808 IS IN THE STATE OF HI
PRESS ANY KEY TO CONTINUE.
```

Looking up telephone area code 808.

So we face a choice. We could do more processing up front to put our data in some kind of order and save processing on our lookups; or we can skip the up front processing and take a little longer on our lookups. For the *STATE/ZIP/AREA CODE* data, the small volume examined each lookup doesn't justify the time and effort required to order it. With a great deal of data, however, the reverse would be true. And somewhere along the data volume spectrum, it's a toss up.

Implicit in this evaluation are such other factors as the human effort required, the size of the program, and its clarity—all of which also affect the programming choice. As with file design, there is never a single right answer.